Journal of Mathematical Sociology
1995, Vol. 20(2-3), pp. 219-246
Reprints available directly from the publisher
Photocopying permitted by license only

RADAR-SOAR: TOWARDS AN ARTIFICIAL ORGANIZATION COMPOSED OF INTELLIGENT AGENTS*

MEI YE

H. John Heinz III School of Public Policy and Management

KATHLEEN M. CARLEY

Department of Social and Decision Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Radar-Soar, an artificial organization composed of intelligent agents, is crafted out of multiple interconnected modules—one of each agent, one of the organizational design (structure plus process), and one of the task. Each agent is based on a sophisticated model of cognition, i.e., Soar. Using Radar-Soar we demonstrate how insight into organizational (and so social) behavior can be gained using computational models that take a meso-approach to organizations.

KEY WORDS: Computational organization theory, organizational design, organizational learning, organizational performance, simulation, artificial intelligence, soar.

Organizations are all crafted, supported and operated by humans. Organizations do not make decisions, people do. Nevertheless, much of our understanding or organizational action, e.g., structuralist, institutionalist, and population ecology, is based on an analysis of organizations that treats as irrelevant, or gives only a brief salute to, the cognitive nature of human action. The information processing perspective (March and Simon, 1958; Cyert and March, 1963; Galbraith, 1973) recognizes the criticality of human cognition to organizational action, but the promise of this approach has not fully materialized (Carley and Prietula, 1994). By intelligent we mean that agents are goal oriented, rational and can accomplish multiple types of tasks by acting on the basis of available information, and thus bringing all the information they have to bear on a problem in order to find its solution. In this paper, we seek to demonstrate how computational organization theory can move us beyond statements of the importance of cognition to a detailed understanding of the organization as a collection of generally intelligent agents.

*This work was supported in part by Grant No. IRI-9111804 by the National Science Foundation.

Radar-Soar, a simulation system that can be used to examine organizational behavior, is crafted out of multiple interconnected modules—one of each agent, one of the organizational design (structure plus process), and one of the task. Each agent is based on a Soar model of cognition, and each agent works on a specific task and acts within an organization with a specific design.

Soar is a model of general intelligence, a unified model of cognition (Newell, 1990). Soar agents are complex adaptive agents that act as humans do, but the types of tasks on which Soar has been shown to act like a human are generally individual in nature. As social scientists, however, we are aware that a wide range of human behavior is social in character. If Soar is a model of human intelligence then it should be able to perform, as humans do, in social or organizational tasks (Carley and Newell, 1990).

Carley, Kjaer-Hansen, Newell and Prietula (1992) demonstrated that Soar can, within limits, act as humans do in a very simple multi-person task. This was a first step in establishing the limits and capabilities of Soar as social agent. In this paper, we take a second step by examining Soar agents with not only knowledge of the multi-agent situation but also knowledge of social roles.

SOAR

Soar (Newell, 1990) is a general purpose language for problem solving. Soar incorporates specific knowledge about the world as a set of rules that guide it in solving problems. Soar learns from experience by remembering how it solves problems. Such learning can occur through a procedure referred to as chunking (Laird, Rosenbloom and Newell, 1986a, 1986b; Rosenbloom, Laird and Newell, 1988) or through model creation (Laird, Newell and Rosenbloom, 1987). Soar is considered to be a unified theory of cognition as it is a single, integrated set of information processing mechanisms that try to explain every aspect of human thought, and not simply one or two experimental results (Newell, 1990).

Soar characterizes all symbolic goal-oriented behavior as search in problem spaces. Such search procedures arguably serve as an architecture for generally intelligent behavior (Laird et al., 1987). Soar's structure is built in levels. The bottom level is memory and as we move up levels we get to decisions and goals. Default knowledge is incorporated into the system as a series of predefined rules (Rosenbloom et al., 1989).

For Soar, search occurs relative to a context. The context is defined by the goal, the problem space, the state, and the available operators. Operators define the set of possible actions given that problem space. The state defines which of the agent's knowledge is currently salient given the current problem space. The goal defines what problem spaces are applicable.

Memory: A general intelligence requires a memory with a large capacity for the storage of knowledge, including declarative knowledge, procedural knowledge, and episodic knowledge. Soar's long-term knowledge is stored in a single production memory, i.e., all knowledge is stored as a series of rules. Each production or rule is a condition-action rule that takes its action when its conditions are met. As these productions are executed memory is accessed. As memory is accessed information

is retrieved from long-term knowledge and placed in the global working memory. The working memory is a temporary memory corresponding roughly to the set of items that are salient at that point in time. When an agent moves from one mental state to another, the current state is kept track of in working memory. One special type of working memory structure is the preference. A preferences encodes control knowledge about the acceptability and the desirability of actions. Acceptability preferences determine which actions should be considered as candidate actions. Desirability preferences define a partial ordering on these candidate actions.

Decision: A general intelligence must have the ability to generate or select a course of action that is responsive to the current situation. In the Soar architecture the next level is the decision level. This level is based on the memory level plus a decision procedure. The decision cycle contains two phases: an elaboration phase and a decision phase. During the elaboration phase, the long-term memory is accessed. All productions that can fire in parallel, and this process continues until no more productions can fire. These productions can set up preferences for either goal context objects (goals, problem-spaces, states and operators) or augment working memory. During the decision phase the preferences for context objects are evaluated. This phase can result in changes in the goal context. The decision cycle lets Soar make its decisions after all the rules have been heard from. Consequently, Soar can use the most powerful knowledge it has available. When there is little knowledge in long-term memory, Soar will behave in the way that resemble general methods such as Hill Climbing (i.e., do whatever seems best at the time) or Means-Ends analysis (i.e., if I am over here and my goal is over there, then I should try to reduce the difference). If Soar has a lot of knowledge in long-term memory and has clear preferences about what to do next then Soar will behave as an expert.

Goals: A general intelligence must also be able to direct its behavior toward some end, i.e., toward an ultimate goal. In the Soar architecture the goal level is based on the decision level. Goals are set whenever a decision cannot be made; i.e., when the decision procedure reaches an impasse. There are four types of impasses: (1) ties; (2) conflicts; (3) no-changes; and (4) constraint failures. When an impasse occurs, Soar will create a subgoal to resolve the impasse, and a new performance context for resolving this subgoal (i.e., a goal = subgoal, problem-space, state, and operators). This automatic subgoaling procedures builds a hierarchical goal structure. A subgoal is terminated when either its impasse is resolved or some higher impasse in the stack is resolved. This unique feature of the architecture is called "universal subgoaling." Universal subgoaling provides a general mechanism for doing conflict resolution. Consequently, Soar can overcome the rigidity of the early expert systems where conflict resolution was done by a fixed mechanism.

Learning: Within Soar learning can occur by the acquisition of chunks. A chunk is a production that summarizes the problem solving that occurs in subgoals. In a chunk, the new rule's conditions are the relevant contents of working memory at the time the impasse arose and its action is the new solution. Chunks acquired at one point in time can be used later to speed up the system's performance. Chunking produces the type of power-law practice curves observed in humans. The power-law practice curves demonstrate that a person's performance on a given task invariably

speeds up as some power of the number of practice trials (the power varies from task to task).

Default Knowledge: Soar has a set of productions that provide default responses to each of the possible impasses that can arise. This default knowledge prevents the system from dropping into a bottomless pit in which it generates an unbounded number of content-free performance contexts (Rosenbloom et al., 1989).

Knowledge Level Computational Models: Within Soar, each agent is first defined in terms of a knowledge level computational model (KLCM). A *knowledge-level computational model* is a class of systems which define an agent behaving in an *environment* (E). The agent is defined as a set of *actions* (A), a set of *perceptual devices* (P), a *goal* (G), and a *body of knowledge* (K). The environment is in one of a set of conditions at each instant of time, and the condition changes over time. The agent interacts with the environment at various times by (1) taking actions and so possibly affecting what environmental condition occurs next, and by (2) using its perceptual devices to acquire knowledge about environment. The environmental condition, the state of the agent, and the interaction between environment and agent determine the joint behavior over time of environment and agent. The agent's goal can affect both the agent's and the environment's behavior, i.e., the agent prefers some joint behavior to others. The agent's behavior is determined by the *principle of rationality*; i.e., if the agent knows that one of its actions will lead to a preferred situation according to its goal, then it will intend the preferred action, and that action will then occur if it is possible (Newell, Yost, Laird, Rosenbloom and Altmann, 1991).

The Knowledge Level Computational Model is a model of goal-directed behavior, because all actions intend to attain the goal of the agent. It is also a model of rational behavior, because everything the agent knows serves the agent's interest. So this model describes a capability for generally intelligent behavior.

Problem Space Computational Models: An agent described at the knowledge level must be realized by a system that has representations of the agents' body of knowledge, and processes that represent new knowledge from perception and determine the actions according to the principle of rationality. In other words, knowledge level system are realized by symbol-level system. In Soar, this means problem spaces. The knowledge level computational model is thus realized by the problems pace computational model (PSCM).

A problem space consists of a set of states and a set of operators. Each operator is applicable to a subset of these states. Applying an operator can change one state to another.

A problem-space system generates *behavior*. Behavior consists of an indefinite sequence of steps. Each step is a state-operator pair. Each step involves two phases: *selecting*, and *applying*. First, the current operator is selected from the operator set. Then the selected operator is applied to current state. If the operator is applicable the current state changes to a new state, otherwise, the current state remains.

A task is formulated using problem spaces by following these steps. The first step is *determine-space*. In this step, a problem space is adopted. Second *goal-setting* occurs. In this step, the desired state is adopted. Next, *initializing* occurs and an initial state is selected. The formulated task is thus accomplished by applying a

series of state-operator pairs. In applying these pairs, the *Independence principle* applies. According to this principle all operators are independent and any operator that can be applied is applied. Finally, termination occurs when the current state reaches the desired state.

Within Soar, if there is a lack of knowledge (the necessary knowledge is not immediately available) in a problem space then another problem space is created to obtain that knowledge. In this new problem space the same procedure for task formulations is followed.

RADAR-SOAR

Radar-Soar is a composite simulation system built out of a series of interconnected modules for task, organizational design, and agent. These modules are designed and interconnected in a three stage process. During the first stage the researcher defines the task. During the second stage the researcher specifies the knowledge and form of the radar-soar agents, that receive as input the task information defined by the researcher and generate opinions. During the third stage, the researcher defines the organizational design by choosing the organizational decision procedure (manager or voting program).

The task defines what knowledge the agents must have and the "parameters" within which the organization must operate. The organizational design defines how many agents are needed, what roles each agent plays, and how the agents communicate.

There is a separate simulation module for each agent in the organization. How many of these modules are used is up to the researcher. Each agent module acts by "observing" something (either information about the task or other agents actions). These observations are done by reading a line from an input file. Each agent module takes some action, e.g., makes a decision or recommendation. This action is "communicated" by writing a line in an output file. Other agents can observe the agent's action, or "hear" what the other agent is communicating by reading this output file. Each problem each agent must solve involves reading from one or more input files and writing to one or more output files.

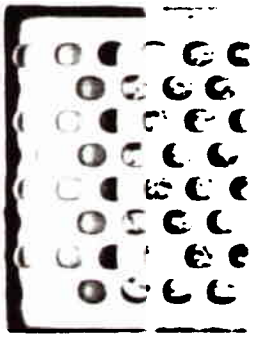
The Radar Task

Choice tasks such as the radar task have been widely studied (Carley and Lin, 1992; Hollenbeck, Sego, Ilgen and Major, 1991). A complete description of the basic choice task we use, the radar task, appears in Lin and Carley (this issue). However, Lin and Carley use a dynamic version of this task where the planes actually move. Here we use a static version where the planes do not move, and the final position of the aircraft is the same as its initial position.

Organizational Design

Organizational design is an extremely broad notion that includes a variety of factors (Thompson, 1967; Scott, 1987). In this paper, we take the stance that organizational design includes the set of formalized relations among people and resources in the

organization, the formal processes by which these relations change, and the formal processes by which people or resources are acquired, let-go, or adapted. We will be particularly concerned here with the organizational structure, the resource access structure, and the training process. We assume that there are two types of agents, analysts and managers. An analyst is an agent who observes features of the aircraft by using radar equipment and makes a recommendation about the state of that aircraft. A manager is an agent who collects recommendations from the analysts and combines these to form a decision about the state of the aircraft. The organizational structure defines who reports to and "commands" or "manages" whom. The resource access structure defines who has access to what resources (including technology and information).



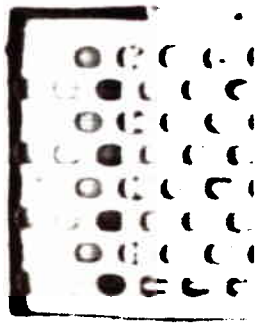
In order to perform a Radar-Soar experiment the user selects the organizational design by making the set of six decisions: Number of analysts? Number of managers? Who does each analyst report to? Which of the features does each analyst observe? How are the agents trained? How extensive is the training. These decisions define the organizational structure, the resource access structure, and the communication structure. In order to simplify exposition, we will focus on a limited set of designs, as described below.

Organizational Structure

We examine organizational structures in which there are nine analysts and the team either has a manager or it reaches its decision through voting. When the team has a manager the organizational decision is made by the manager. In this case, each analyst passes its decision to the manager who collates this information and comes to a final decision. In contrast, when the team votes the organizational decision is the majority decision made by the nine analysts. We focus on these team structures as earlier work by social network theorists and organizational theorists suggests that the presence or absence of a manager may be critical to the organization's performance.

Resource Access Structure

We examine two resource access structures: distributed and blocked. In both the blocked and distributed structure each analyst sees information on only three of the nine features of the aircraft. In the blocked structure three analysts have identical mental models (at least based on incoming information); i.e., three analysts see exactly the same three pieces of information. In the distributed structure no two analysts see exactly the same information, although each piece of information is seen by all three analysts. Which three features the analyst has information on depends on the analyst's position in the resource access structure.



Training Process

All agents are trained minimally to make decisions on the basis of their experience. Each agent sees a sequence of 10 aircraft, makes a recommendation, and then is told the true state of each aircraft.

Organizational Performance

Overall performance is measured as the percentage of "correct" decisions made by the organization. An organization's decision is correct if its classification of the aircraft exactly matches the true nature of the aircraft (e.g., deciding that the aircraft is friendly when it is actually friendly). A second measure is severity of error. An organization can make an error by being either "off by one" (e.g., deciding that the aircraft is friendly when it is actually neutral) or "off by two" (deciding that an aircraft is friendly when it is actually hostile or vice versa). The severity of error is defined as percentage of total errors that are severe.

We also examine cumulative performance which is the percentage of total decisions that have been made correctly at this point in time. Similarly, we examine cumulative severity of error.

Performance measures are calculated outside the simulation using post-processors. For the team with voting a post-processor extracts from each of the nine analyst's output files the analyst's decision for each of the observed problems. The majority is calculated and this becomes the organization's decision. For the team with manager a post-processor extracts the manager's decision for each problem from the manager's output file. The manager's decision is the organization's decision.

Agents

We employ agents in two roles—managers and analysts. We will describe the Radar-Soar agents in three steps. First, we will define the knowledge and actions of the analyst and manager. Second, we will describe the knowledge level computational model (KLCM) for the analyst and manager. Finally, we will describe the communication procedure between the Soar agents.

STEP 1 Knowledge and Actions Both analyst's and manager's have various knowledge about how to communicate and how to read reports. What is critical to their actions, however, is their initial "task" knowledge.

The initial knowledge of each analyst about aircrafts can be stated as three rules.

1. If all the observed features are low, then decide that the aircraft is friendly.
2. If all the observed features are medium, then decide that the aircraft is neutral.
3. If all the observed features are high, then decide that the aircraft is hostile.

The manager's initial knowledge about the decisions their subordinates make also can be stated as three rules.

1. If all subordinate analysts report that the aircraft is friendly, then decide that the aircraft is friendly.
2. If all subordinate analysts report that the aircraft is neutral, then decide that the aircraft is neutral.
3. If all subordinate analysts report that the aircraft is hostile, then decide that the aircraft is hostile.

Both the analyst's and manager's task knowledge accumulates over time relative to feedback from the external environment regarding the true state of the aircraft.

This feedback is identical for all agents regardless of their position in the organizational structure. When the current situation is beyond the agent's initial knowledge the agent will make a decision by reasoning, and not simply by guessing.[1]

The actions taken by all analysts are: observe air-space; analyze the information observed from the air-space; make a decision about the aircraft to tell whether it is hostile, neutral, or friendly based on current knowledge; report each individual decision; get feedback on the true state of the aircraft. If the analyst is in a team with a manager then the analyst has an additional action: receive command from manager. Further, when the analyst is in a team with manager then the report action is modified so that the report goes to the manager.

The actions taken by the manager are: propagate commands like "observe air-space" and "tell me" (your decision); receive individual decision from each subordinate; make organizational decision; get feedback on the true state of the aircraft.

STEP 2 Knowledge Level Computational Models The second step involves specifying the knowledge level computational model (KLCM) for both the analyst and the manager. This KLCM is realized by developing the problem space computational model (PSCM) for each analyst and for the manager. We begin by describing the KLCM for the analyst and then the manager. We then provide general information on PSCM's. We then describe the analyst's and manager's PSCM.

The KLCM implemented for each analyst is shown in Table 1. The KLCM implemented for the manager is shown in Table 2. Both types of agents operate in the same radar environment and have the same overall goal. For each analyst, their initial task knowledge consists of three rules denoting that if the inputs are consistent then choose the corresponding answer. The manager has the same type of initial task knowledge but differs in that it looks not at features of the aircraft but at decisions made by analysts. The manager can command the analysts, while the analysts can only report to the manager.

STEP 3 Problem Spaces Computational Models As previously noted, an agent described at the knowledge level must be realized at the symbol level. This is done by describing the problem spaces of the agents.

Each analyst is implemented using 11 problem spaces. These 11 problem spaes and the hierarchical connections among them are shown in Figure 1. The top problem space contains the initial state and the desired state (the goal state) for each analyst. The communication problem space consists of three operators. The first is the get-command operator which receives one command at a time from the manager. This operator is not used when the agents are in a team with voting. The second is the report operator which reports the decision the agent made about the aircraft. The third operator proposed is get-feedback which is the initiating action by the agent to retrieve the feedback information and update its knowledge. In addition, for the manager there are two kinds of commands/operators: "observe" and

[1]For example, if the first aircraft an agent sees has the features not all high, all low, or all medium, then the agent has no knowledge of what status this aircraft is. What the agent will do is to compare the features of the current aircraft to the models (initial knowledge) in its repertoire, then the agent will choose the model which has the closest match to the current aircraft.

TABLE 1
Knowledge Level Computational Model for Analyst

Environment	Stylized radar station
Goal	Make organizational decision
Knowledge	Initial Knowledge if feature – 1 = ··· feature – 3 = friendly then organizational decision = friendly if feature – 1 = ··· feature – 3 = neutral then organizational decision = neutral if feature – 1 = ··· feature – 3 = hostile then organizational decision = hostile
Problem Space	Interpret command first—"Observe" airspace second—"Report" decision wait for all radar information make decision report decision get feedback add knowledge
Action	Interpret command Make decision Report decision Get feedback

TABLE 2
Knowledge Level Computational Model for Manager

Environment	Stylized radar station
Goal	Make organizational decision

Knowledge	Initial Knowledge
	if agent – 1 = ··· agent – 9 = friendly then
	organizational decision = friendly
	if agent – 1 = ··· agent – 9 = neutral then
	organizational decision = neutral
	if agent – 1 = ··· agent – 9 = hostile then
	organizational decision = hostile

Problem Space	Propagate command first—"Observe" airspace second—"Tell me" decision wait for all analyst's decisions get all analyst's decisions make organizational decision based on analyst's decisions get feedback add knowledge send command "Task finished" to analysts
Action	Propagate commands Receive responses Make decision Get feedback

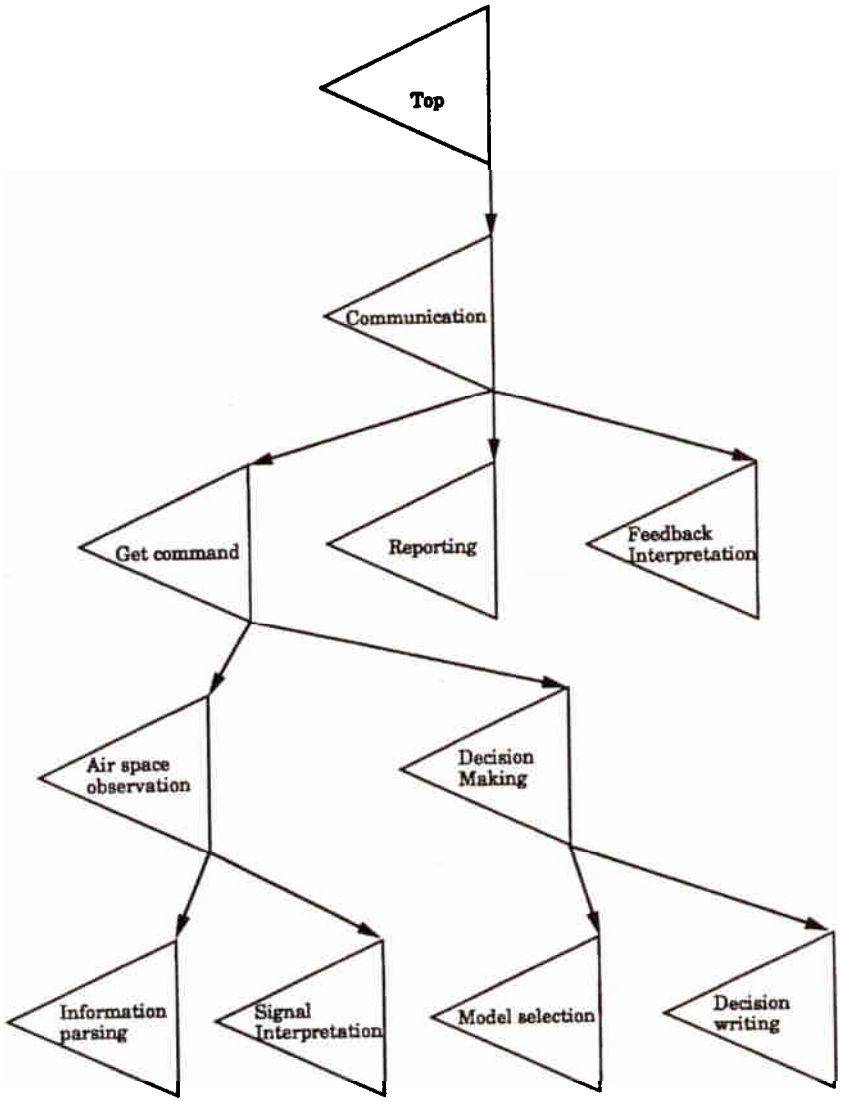


FIGURE 1. Problem space computational model for analyst.

"tell me." These commands cause an analyst to move to two different problem spaces, the observe air-space problem space and the make decision problem space. The observe air-space problem space includes the parse information operator, which is used by the agent to analyze the signal captured from the air-space about the flying object, and the interpret operator, which converts each signal to the attribute it represents. The make decision problem space consists, first of a model select operator which compares the information of the aircraft to all the different models in an agent's mind and then makes a decision based on the maximum match. Second,

there is a write decision operator which allows the agent to track its decision so that it will be able to report this decision to the manager later.

We have been describing Radar-Soar in the fashion that Soar models are typically described. It is also useful to look at Radar-Soar from a standard flow chart perspective. In Figure 2 we have overlaid the flowchart with boxes outlined by dashed lines, where each box indicates which processes are within which problem space.

The analyst (see Figure 2) begins in the communication problem space and from there subgoals to any of the following problem spaces: get command, report, or feedback. However, the agent is in an organization that constrains its access to information and therefore actions. These constraints enable the agent to first choose to get a command. Then the agent can subgoal to either the airspace observation or the make decision space. Which space the analyst moves into depends on the available information. If the analyst has received an observe command then it subgoals to the air space observation problem space. Alternatively, if it has received the report command it subgoals to the decision making problem space. If the agent is observing the air space then it can subgoal to either the information parsing or signal interpretation problem space. However, the agent has prior knowledge embodied as preferences that enables it to choose to first parse the information and then to interpret the signal. Interpretation of the signal results in satisfying that subgoal and agent then "pops up" to the get command problem space.

At this point the agent must again choose whether to move into the observe or the decision space. However, it now has new knowledge that it has an observation, and this knowledge is used to eliminate observation as an action possibility. When the agent subgoals to the make individual decision problem space it can subgoal further to either the model select or make decision subspace. Knowledge in the form of preferences forces the agent to first select a model and then to make a decision. After making a decision the agent pops back to the communication problem space. Now that the agent knows its decision, reporting is possible. Thus the agent subgoals to the report problem space. After the agent has filed its report it pops back up to the communication space again. Now it has the information that it has filed a report, and the only action remaining for which there is an active preference is to get feedback. The analyst now subgoals to the feedback problem space. In the feedback problem space the agent first determines if feedback is available, and if so compares its answer with the correct answer. If the agent's answer is not correct it creates a new model, if it is correct it increases ht eprobability of the current model (by adding a duplicate of the current model to the set of the available models the probability of a model is set by the number of available models that are identical). After either finding that there is no feedback, or after having processed the feedback the agent then pops back to the communication problem space. On returning from the feedback problem space the agent has new knowledge. This new knowledge enables causes the agent to prefer to get a command over reporting or getting feedback.

Because of the limits of human intellective capacities in comparison with the complexities of the problems that individuals and organizations face, the problem solving processes for humans are boundedly rational (March and Simon, 1958). This rational behavior calls for simplified models that capture the main features of a

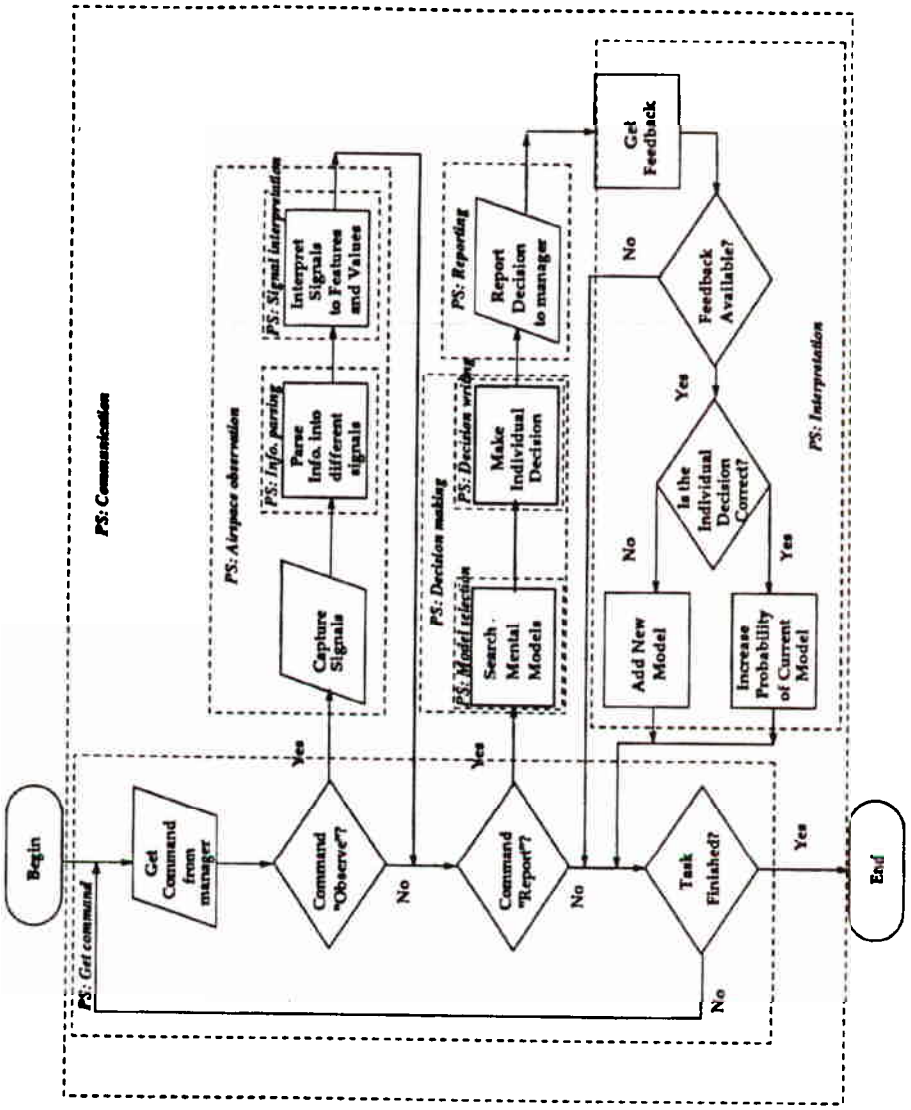


FIGURE 2. Flow chart for analyst.

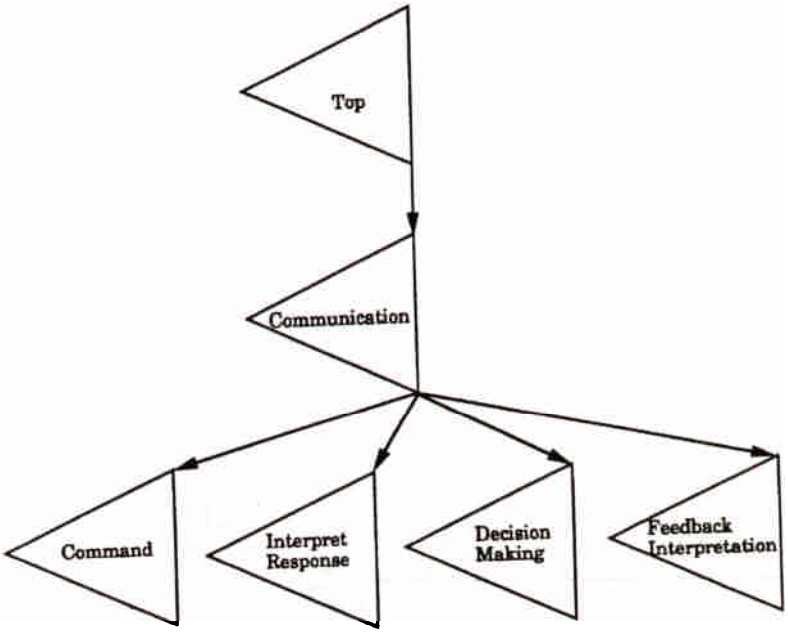


FIGURE 3. Problem space computational model for manager.

problem while eliminating its complexities. Radar-soar agents simulate this behavior by randomly choosing among the best matched models to the current situation. To find the best matched model the radar-soar agent first compares the models based on feedback with the characteristics of the current aircraft. Then the agent counts the number of features that match between model and the current aircraft. The radar-soar agent makes a final decision by choosing the model with the maximum number of matches. This model predicts the status of the aircraft (i.e., friendly, neutral, or hostile). This is the basic decision making process in radar-soar. This process is used whether the agent is an analyst or a manager.

The manager agent is implemented using six problem spaces which are hierarchically connected as shown in Figure 3. The top problem space contains the initial state and the desired state (goal state) for the manager. The next problem space is the communication problem space which sequentially proposes four operators: the give command operator; the receive response operator; the make organizational decision operator; and get feedback operator. These four operators in turn propose four problem spaces.

The manager takes a series of actions as though it were operating concurrently with the analysts (Figure 4). The models that have the best match are preferred. The manager then randomly chooses among the best matching models. The chosen model defines the decision. The manager then gets feedback and if the decision made was incorrect the manager creates a new mental model with this case and the appropriate feedback.

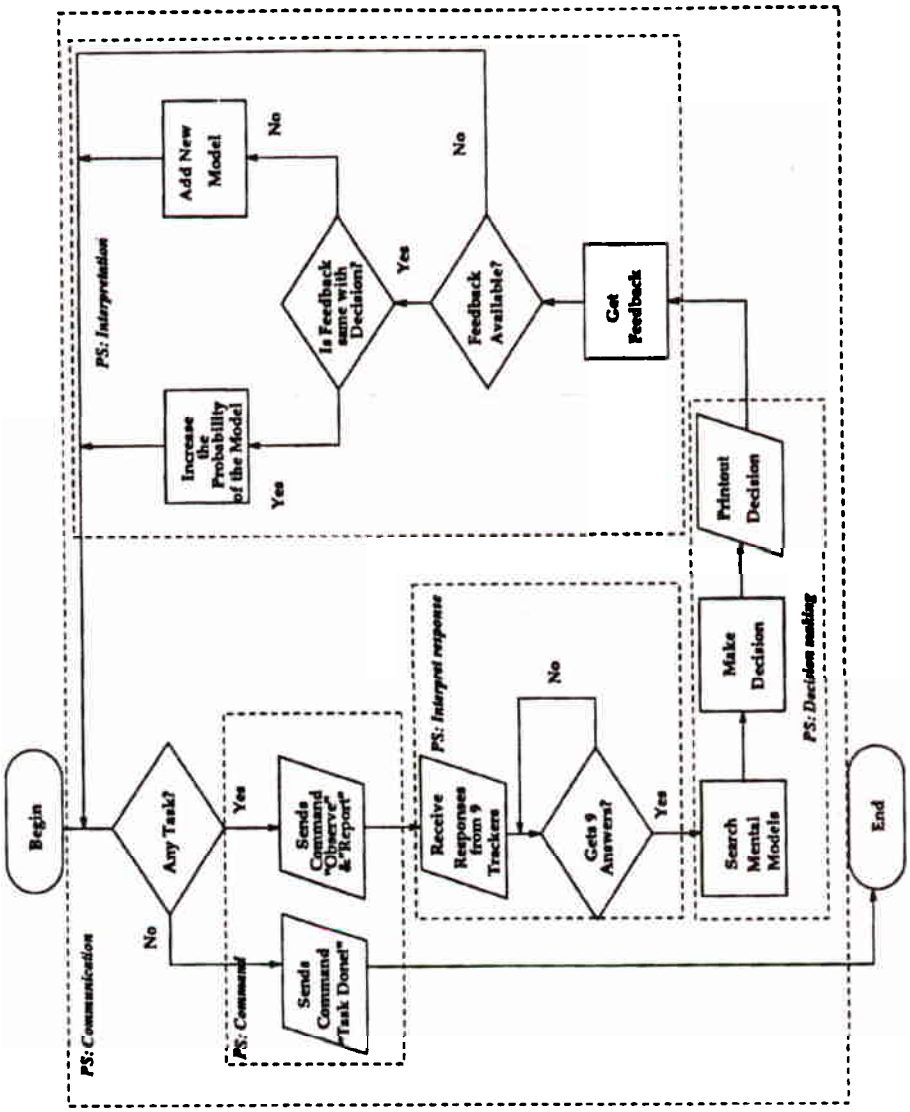


FIGURE 4. Flow chart for manager.

In Figure 4, we see that the manager begins in the communication problem space. From the communication problem space the manager can subgoal to any of the following problem spaces: give command, receive response, make organizational decision, or feedback. The organization constrains the agent's access to information. The information available to the agent causes it to choose to give a command. Specifically, first the manager checks to see if there is an item in the airspace. If so, the manager commands the analysts to either examine the aircraft or report their decision if there are no more aircraft in the airspace (i.e., no more lines in its input file). If the task is not finished the manager sends a pair of commands, first "observe" then "report". After having sent this pair of commands the manager pops back to the communication space. Now the manager has information that a command has been sent. The manager's preferences now cause it to subgoal to the receive response problem space. In this space the manager waits for all nine analysts responses. The manager reads and stores the analysts' responses. After all analysts have responded the manager has as information the responses of the analysts. After it has all responses the manager pops back up to the communication space. Then the manager subgoals to the decision making problem space. The manager first searches its mental models for the best match and then makes a decision. The manager then pops back to the communication problem space and then subgoals to the feedback space. In the feedback space the manager follows a process similar to that of the analysts.

In the search mental models step the manager selects and evaluates all of its mental models effectively in parallel (Figure 5). A model is a description of a situation and a decision. Each model is represented as a production rule of the form if the aircraft has characteristics *a*, *b*, and *c* then decision should be that the aircraft is *x* (friendly, hostile, or neutral). Once a model is selected the manager determines how well the model matches the observed aircraft. The more of the nine features that are the same, the better the match.

Once the match has been determined for each model, the manager then makes a decision (Figure 6). First, the manager selects all models in parallel. Second, the manager sets the preference for each model in parallel. Note, associated with each model is a preference. This preference indicates the degree to which the model is acceptable to the agent. These preferences are based on the extent to which the model matches the characteristics of the aircraft currently in the airspace. The better the match the higher the preference for that model.

The models are ordered by the strength of the preferences. If there is a model that is most highly preferred then the decision suggested by the model becomes the manager's decision. Otherwise, the manager chooses a model from among those models that are highly preferred. Which model is chosen is determined randomly. However, since the manager stores a model for each problem that it observes this procedure generates the result that the manager's decision has a probability associated with it proportional to the number of times the manager has observed an aircraft of this type with this outcome. For example, imagine that there are 10 models in the manager's knowledge base. Let one of these models have a match of 3, three a match of 5, and six a match of 8. In this case, no model matches perfectly (no match of size 9). However there are six models that are highly and equally preferred. Imagine that

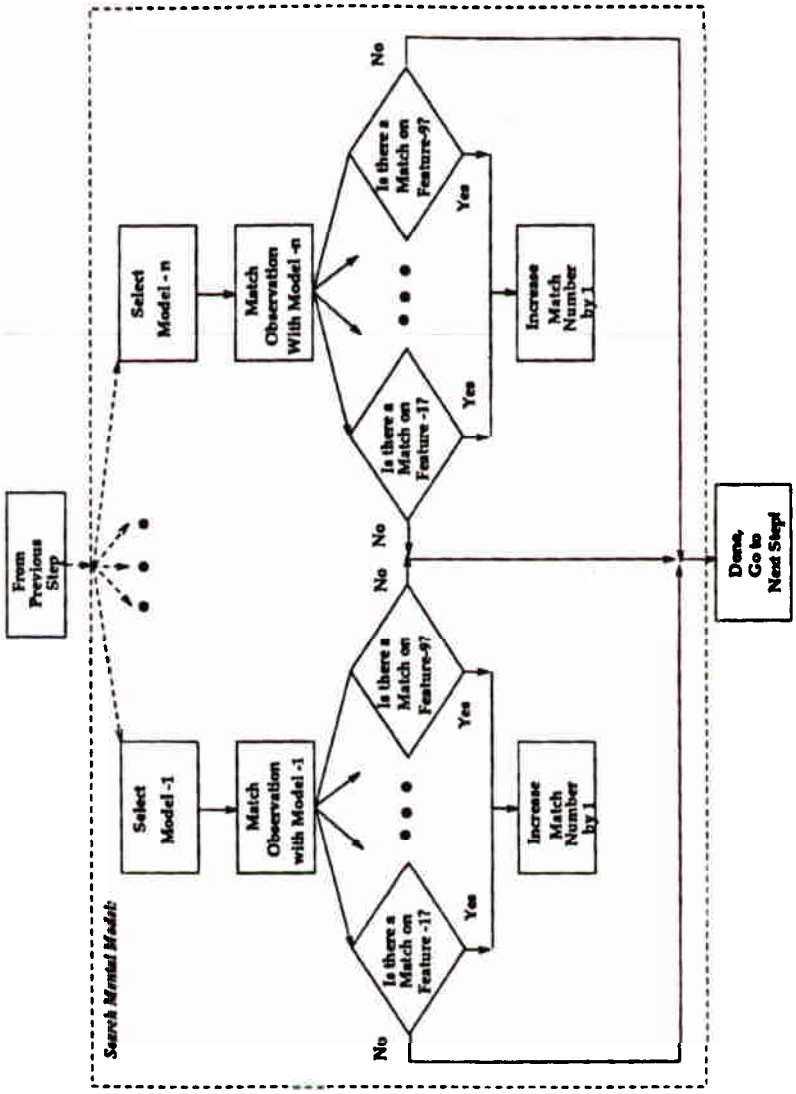


FIGURE 5. Flow chart for search mental model.

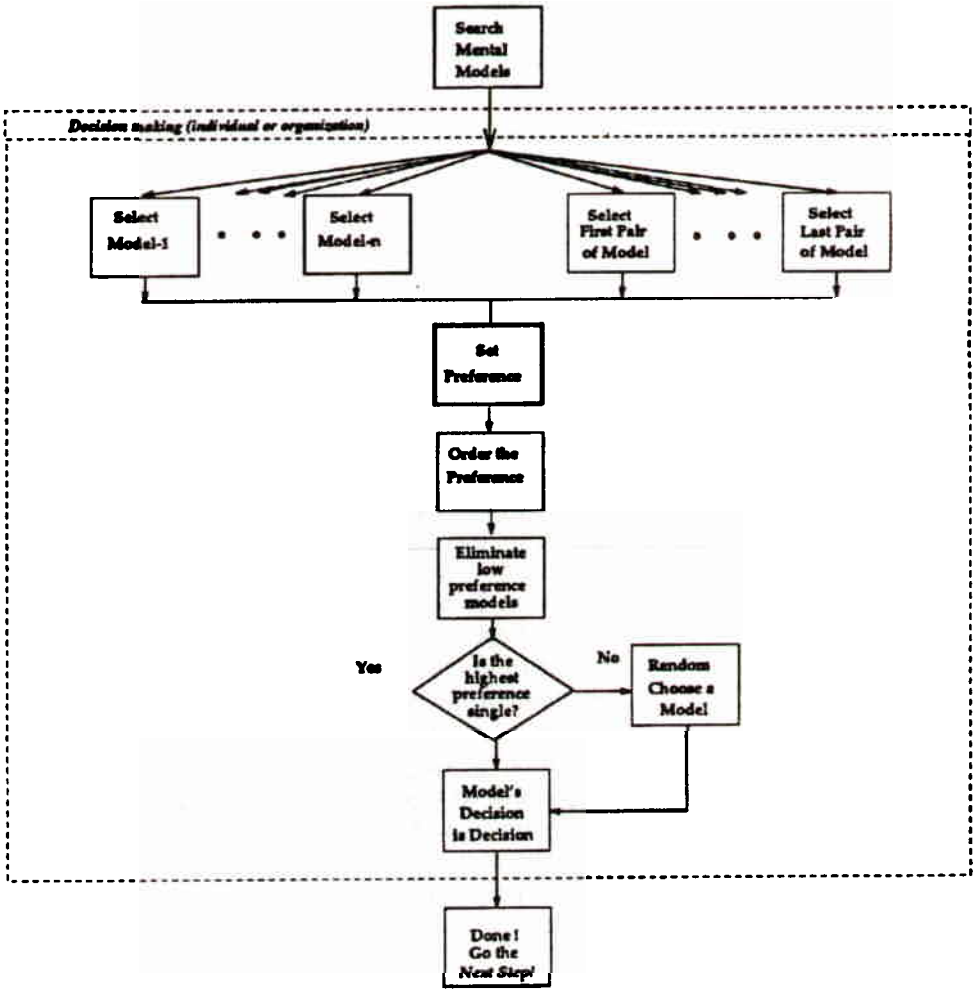


FIGURE 6. Flow chart for decision making.

of these six models, three indicate that the aircraft is friendly, two indicate that it is neutral, and one indicates that it is hostile. The manager eliminates the low preference models. Now the manager randomly chooses from the remaining six models. Half of the time the manager will state that the aircraft is friendly, one-third of the time the manager will state that the aircraft is neutral, and one-sixth of the time the manager will state that the aircraft is hostile. As the manager observes more aircrafts these frequencies will change.

STEP 4 Communication Communication between agents is done simply by coordinating the contents of the input and output files for the agents. As noted previously, the organizational design and task define the content of these files.

The first 10 lines from an input file for an analyst is shown in Table 3. A full input file has 60 lines. Each line indicates that the manager has commanded the tracker to

TABLE 3
Analyst's Input

Aircraft id	Order	Feature	Value	Feature	Value	Feature	Value
3	1	f1	1	f4	1	f7	1
234	2	f1	1	f4	1	f7	2
459	3	f1	1	f4	2	f7	3
2036	4	f1	1	f4	3	f7	2
5131	5	f1	1	f4	1	f7	1
5483	6	f1	1	f4	2	f7	1
7884	7	f1	2	f4	3	f7	3
9842	8	f1	2	f4	2	f7	2
12614	9	f1	2	f4	1	f7	1
2	10	f1	1	f4	1	f7	1

examine the environment. Each line contains information on three features of the aircraft present at that time in the environment. The analyst reads one line, parses it, interprets it, and makes a decision. This process continues until all lines have been read.

While running each agent creates a "trace." The trace is the sequence of actions taken by the agent. Each numbered line indicates a decision cycle. Following is a trace of the first 108 decision cycles for the analyst who received the input file shown in Table 3. The decision cycle is denoted by the number prior to the colon at the beginning of a line. Unnumbered lines (such as "open File Done!" are the result of an I/O operation on the part of the agent and track specific radar-soar actions. In this trace, G is for goal, P for problem space, S for state, and O for operator. A $\Rightarrow$ indicates that the system is subgoaling. Indentation indicates the depth of the subgoal. Outdentation occurs when the agent completes a subgoal and pops back to a previous space. During decision cycle 27 he analyst prints out the characteristics of the observed aircraft. From decision cycle 39 to decision cycle 95 the analyst is counting the degree of match for each model in its repertoire. Then it sets its preference. Unnumbered lines after 96 show the content of the best matched model.

```
Output of a tracker
Soar>
     0: ==>G: G1
     1:    P: P1 (top-ps)
     2:    S: S1 (top-state)
Sun Jul 11 17:14:33 EDT 1993

open File Done !

 (Agent) Hostile Model Done !

 (Agent) Friend Model Done !
```

```
(Agent) Neutral Model Done !

    3:     O: O1 (communication)
    4:     ==>G: G2 (operator no-change)
    5:        P: P2 (communication-ps)
    6:        S: S1 (top-state)
    7:        O: O3 (get-command)
    8:        ==>G: G3 (operator no-change)
    9:           P: P3 (get-command-ps)
   10:           S: S1 (top-state)
command =  observe

   11:           O: O5 (observe-air-space)
   12:           ==>G: G4 (operator no-change)
   13:              P: P4 (observe-air-space-ps)
   14:              S: S1 (top-state)
 test-info =  3

   15:              O: O6 (parse-command)
   16:              ==>G: G5 (operator no-change)
   17:                 P: P5 (parse-command-ps)
   18:                 S: S1 (top-state)
Parse command finished !

   19:              O: O7 (interpret)
   20:              ==>G: G6 (operator no-change)
   21:                 P: P6 (interpret-ps)
   22:                 S: S1 (top-state)
   23:        O: O4 (report)
   24:        ==>G: G7 (operator no-change)
   25:           P: P7 (report-ps)
   26:           S: S1 (top-state)
current: object ID = 3 attribute = speed value = low

current: object ID = 3 attribute = altitude value = low

current: object ID = 3 attribute = identification value = low

   27:        O: O8 (get-command)
   28:        ==>G: G8 (operator no-change)
   29:           P: P8 (get-command-ps)
   30:           S: S1 (top-state)
command =  tell

   31:           O: O10 (make-decision)
   32:           ==>G: G9 (operator no-change)
   33:              P: P9 (make-decision-ps)
   34:              S: S1 (top-state)
   35:              O: O11 (compare-to-model)
   36:              ==>G: G10 (operator no-change)
   37:                 P: P10 (compare-to-model-ps)
```

```
  38:                S: S1 (top-state)
  39:                O: O14 (accumulate)
  40:                O: O18 (count-item)
  41:                O: O13 (accumulate)
  42:                O: O29 (count-item)
  ...
  90:                O: O12 (accumulate)
  91:                O: O149 (count-item)
  92:                O: O14 (accumulate)
  93:                O: O150 (count-item)
  94:                O: O12 (accumulate)
  95:                O: O14 (accumulate)
  96:                O: O146 (set-preference)
final status ==  friend  match-number ==  3  model ==  F1
type ==  friend  Attribute ==  speed  Value ==  low
type ==  friend  Attribute ==  range  Value ==  low
type ==  friend  Attribute ==  direction  Value ==  low
type ==  friend  Attribute ==  altitude  Value ==  low
type ==  friend  Attribute ==  angle  Value ==  low
type ==  friend  Attribute ==  corridor-status  Value ==  low
type ==  friend  Attribute ==  identification  Value ==  low
type ==  friend  Attribute ==  size  Value ==  low
type ==  friend  Attribute ==  radar-emission-type  Value ==  low
  97:              O: O153 (feedback)
  98:              ==>G: G11 (operator no-change)
  99:                 P: P11 (feedback-ps)
 100:                 S: S1 (top-state)
feedback number === 30

 101:              O: O155 (write-decision-slot)
 102:              ==>G: G12 (operator no-change)
 103:                 P: P12 (write-decision-slot-ps)
 104:                 S: S1 (top-state)
 105:        O: O9 (report)
 106:        ==>G: G13 (operator no-change)
 107:           P: P13 (report-ps)
 108:           S: S1 (top-state)
. .
```

After the analyst has executed all commands (read and processed all lines from the input file), the tracker stops. The Radar-Soar agent then prints out a message such as that shown below.

```
command =  exercise-finished
Sun Jul 11 20:23:54 EDT 1993
All files have been closed !
  Goal top-goal succeeded.
 goal top-goal achieved
System halted.
Soar>
```

The particular analyst, for whom we showed the input and the first part of its trace, completed its job in 15355 decision cycles and 30446 elaboration cycles. During this time it fired 531456 productions, but used only 197 unique productions in evaluating all commands in the input file. Of these 197 productions, 99 were default and 98 were developed specifically for radar-soar. The large number of fired relative to unique productions is indicative of the highly repetitive nature of this task. These performance characteristics are similar across all analysts.

Each agent also generates an output file of 60 lines that simply records which aircraft it saw and what it's decision was.

The manager then takes the output files of the nine analysts as input. The manager generates, as output, a trace, similar to that for the analyst. In addition, the manager generates an output file identical in form to the analysts listing the aircraft id and the decision.

EXPERIMENTAL DESIGN—DATA DESCRIPTION

We examine organizations with four designs: team with voting, blocked resources; team with voting, distributed resources; team with manager, blocked resources; team with manager, distributed resources. First each agent is trained on 10 cases. The agents do not stop learning after the training period. They continue to learn from their experience. Then each agent in each organization sees information on a sequence of 60 aircraft. These 60 cases are divided into two phases, each of which contains 30 cases. During the first phase the agents receive feedback, and during the second phase there is no feedback. Agents can only receive feedback after they make their decision on the aircraft.

The overall design of this research is such that there are organizations, each of which faces 60 different aircraft, the first 30 for which there is feedback and the last 30 for which there is no feedback. These organizations vary in organizational structure and resource access structure.

RESULTS

Prior research suggests that decentralized structures like a team with voting are flexible structures that can respond rapidly (Aldrich, 1979) and provide more effective communication for single-problem learning (Wilensky 1967; Simon, 1973; Galbraith, 1973). In contrast, more centralized structures, such as a team with manager, are expected to take longer to make decisions but to be more resilient in the face of difficulties. While less has been written in the organizational literature on shared mental models we might expect that those organizations where the individuals have a common view (blocked structure) will tend to outperform those where each individual has a distinct view (distributed). In Table 4, we see that as expected the more decentralized structures, the team with voting, exhibits higher performance. In addition, those organizations where more individuals shared their mental models (blocked) outperformed those with the same organizational structure but a more distributed resource access structure.

TABLE 4
Overall Radar-Soar Performance Measured as the Percentage of All Cases That the Organization Successfully Classified

Oranizational Structure	Resource Access Structure	
	Blocked	Distributed
Team with Voting	71.67%	68.33%
Team with Manager	61.67%	60.00%

$N = 60$ for all cells.

TABLE 5
Overall Radar-Soar Severe Errors Measured as the Percentage of the Errors Made by the Organization That Were Severe Across All Cases

Oranizational Structure	Resource Access Structure	
	Blocked	Distributed
Team with Voting	7.14%	10.52%
Team with Manager	8.69%	8.33%

$N = 60$ for all cells. N is the number of cases.

On the one hand, these results seem straightforward. On the other hand, they fly in the face of some previous research. For examples, previous research suggests that teams without managers must be explicitly coordinated to perform effectively (Durfee, 1988; Durfee and Lesser, 1988). In Radar-Soar, final decisions made by voting (the majority rule) involve an implicit coordination scheme with little cost. These experiments show that the voting team outperforms the team with manager. This may be due to the Radar-Soar agents acting as novices. Future work will need to examine whether or not organizational performance increases with the expertise of the manager. As another example, previous research also suggests that the equal allocation of resources and effort, as in the case of the blocked structure, may not be the optimal coordination strategy (Arrow and Radner, 1979). In contrast, this research suggests that when the agents exhibit rigid adaptive learning and probabilistic decision making behavior, such blocked structures are slightly better than more distributed structures. Both of these examples suggest that part of the value of the Radar-Soar system is the ability to look at organizations that differ in design.

The theories, and the results in Table 4, only speak to general performance; neither addresses degree of accuracy. In Table 5 we see that organizational accuracy varies with the organizational design. However, the relationship between severe errors and organizational design is complex. The team with voting and a blocked structure makes the fewest errors (Table 4) and very few of these are severe (Table 5). In contrast, the team with manager and a distributed structure makes the most errors (Table 4) but also makes relatively few severe errors (Table 5).

Next we examine the impact of feedback. In general, organizations perform better given feedback if the agents can learn (Carley, 1991). To the extent that organizational behavior is largely symbolic (Feldman and March, 1981) then feedback may not improve performance. Further, we might expect that whether or not there is

TABLE 6
Impact of Feedback on Radar-Soar Performance Measured as the Percentage of Cases That the Organization Successfully Classified

Organizational Structure	Feedback Resource Access Structure		
	Blocked	Distributed	Across Resources
Team with Voting	73.33%	63.33%	68.33%
Team with Manager	63.33%	53.33%	58.33%
Across Teams	68.33%	58.33%	63.33%
		No-Feedback	
	Blocked	Distributed	Across Resources
Team with Voting	70.00%	73.33%	71.61%
Team with Manager	60.00%	66.77%	63.38%
Across Teams	65.00%	70.05%	67.50%

$N = 30$ in for the central cells.
$N = 60$ in the Across Team rows and Across Resources columns.
$N = 120$ in the Across Team and Across Resource cell.
N is the number of cases.

TABLE 7
Impact of Feedback on Radar-Soar Severe Errors Measured as the Percentage of the Errors Made By the Organization That Were Severe

Organizational Structure	Feedback Resource Access Structure		
	Blocked	Distributed	Across Resources
Team with Voting	0.00%	18.18%	10.53%
Team with Manager	0.00%	7.14%	4.00%
Across Teams	0.00%	12.00%	6.81%
		No-Feedback	
	Blocked	Distributed	Across Resources
Team with Voting	11.11%	0.00%	5.88%
Team with Manager	16.67%	10.00%	13.64%
Across Teams	14.29%	5.56%	10.26%

$N = 30$ in for the central cells.
$N = 60$ in the Across Team rows and Across Resources columns.
$N = 120$ in the Across Team and Across Resource cell.
N is the number of cases.

feedback the relations between organizational design and performance will remain as described above.

Our analysis, however, shows a slightly different pattern. In Table 6 we see that there are organizational designs where performance is better when there is no feedback than when there is feedback. We cannot take these results, however, as supporting the Feldman and March claim of symbolic action. Rather, it is important to keep in mind that the Radar-Soar agents continue to learn from experience, whether or not there is feedback. This learning may be superstitious, but it can, nonetheless lead to improved performance. However, if the agents are engaged in superstitious learning then, in an environment without feedback, we would expect to see a decrease in accuracy and an increase in severe errors as the agents face more tasks. Turning to Table 7 we see that this is indeed the case.

When feedback is available we get the expected relation between organizational design and performance. In contrast, when there is no feedback we observe a different relation between design and performance. Under both the feedback and no feedback condition, distributed structures (team with voting) outperform centralized structures (team with manager). Shared mental models (which occurin blocked structures) are an advantage when there is feedback, but a disadvantage when there is no feedback. In Table 6 we see that organizations with distributed resource access structure, where each individual has a slightly different view of the organization, not only do beter than blocked structures but do better than they did when they had feedback. In Table 7 we see that while distributed resource access structures are more prone to severe errors when there is feedback, they are prone to fewer severe errors when there is no feedback. Further, the centralized team with manager structure makes fewer severe errors than the decentralized team with voting when there is feedback. However, the opposite is the case when there is no feedback. The presence or absence of feedback can affect the relative performance of the different organizational designs.

Now let us consider the performance of Radar-Soar over time. Figure 7 shows the performance of Radar-Soar, under different organizational designs using the measure of cumulative performance. Initially agent performance is highly variable. As agents learn through feedback, their performance begins to stabilize. Regardless of organizational design performance is around 60%. After feedback is discontinued (time 30) performance stabilizes. We also see that organizations with blocked structures tend to learn the fastest. And that, teams tend to do better in the absence of feedback.

DISCUSSION

The procedure we have outlined for examining organizational behavior using Soar agents could be used to examine organizational designs and tasks other than those that we have described. In order to do so, the researcher would simply need to establish different organizational structures, resource access structures, and/or communication structures. Alternative tasks could be examined by using different defining rules for characterizing the true state of the aircraft.

Running such simulations requires substantial computer resources. In our experiments, we have only 60 cases for each type of organizational designs as compared to the total possible cases of $3^9 = 19683$. This limitation is due to the computer facilities and time constraints for running the radar-soar experiment. Each analyst needs approximately three hours (10825.297 seconds) of CPU time to finish an experiment consisting of 60 aircraft, 30 with and 30 without feedback. It takes more than three hours (11738.645 seconds) for the manager to finish the same experiment. In contrast, humans can finish this experiment in about 30 minutes regardless of whether they are analysts or managers.

Soar agents (analysts) need almost four times longer than humans to reach the same decision for several reasons. First, humans (unlike the radar-soar agents examined) cannot actually remember all the problems that they have seen (all aircraft and its associated feature set and true state). Thus humans often make decisions us-

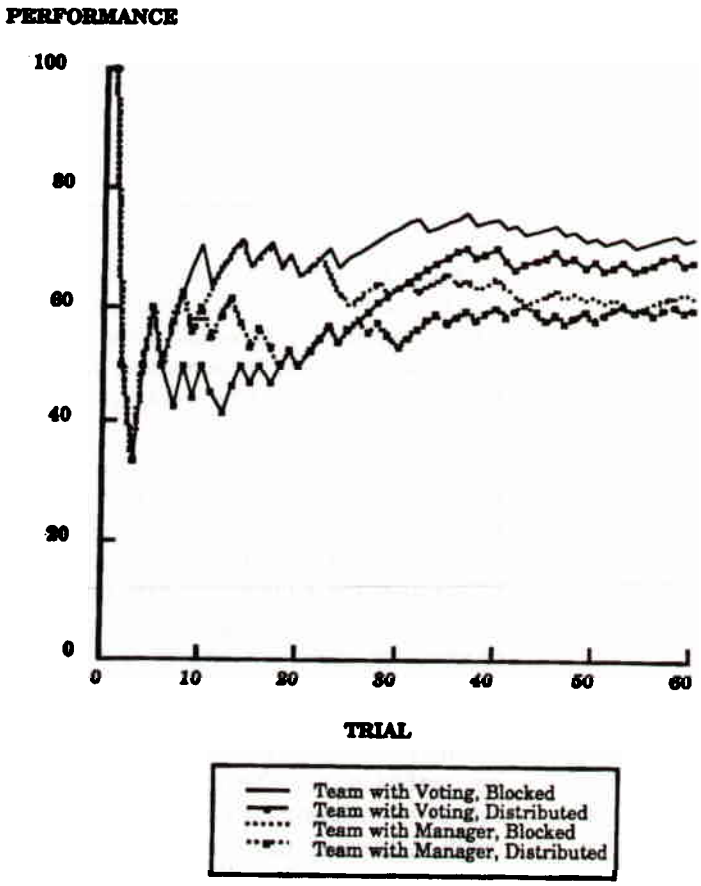


FIGURE 7. Radar-Soar performance over time.

ing processes such as generalization and partial recall. Such processes speed by the decision making process. While these processes could be modeled in Soar we have not done so. Radar-soar agents on the other hand, remember all problems.[2] Further, when the radar-soar agent wants to compare the current situation with the model in their cognition, that radar-soar agent has to propose operators for comparison, and for counting the matched features. How many models the Soar agent has and how many features in each model determines how many compare operators and how many match operators are proposed. Since each operator represents a single action, the radar-soar system has to apply procedence or indifference preferences among these operators. Establishing a preference ordering also takes time. For example,

[2]The learning model used in Radar-Soar is based on assumptions which reflect the observation that organizational behavior is historically based (Carley, 1992). Organizations rely on experience, incrementally adapting their response to similar problems as they receive feedback on their previous decision (Lindblom, 1959; Steinbruner, 1974; Levitt and March, 1988). Thus, each Radar-Soar agent is treated as perfect historian.

for a radar-soar manager, if there are 60 models and each model has nine features, the total number of comparison and match operators would be $60 \times 9 = 540$. To establish preferences each two of these models must be compared, thus resulting in 145530 comparisons. Thus, as the number of models grows the time to make a decision slow down. This behavior of the radar-soar agents is at odds with what we know of human behavior, in that it does not follow the power-practice law.

In theory, we might have observed the power-practice law had we turned on chunking. In our experiments, we set the chunking switch off, because first, turning the learning switch on in Soar needs more restrictive coding.[3] Second, learning through chunking only comes into effect when the same situation appears again, which is not true for the particular 60 cases in our experiment. Whether alternative representations would allow partial learning through chunking is a question for future research. In particular, future research should consider alternative representations of the radar-tracking problem in Soar that would avoid these problems.[4]

We could, and have, coded radar agents without using Soar. These include the CORP (Carley and Lin, 1995) and the DYCORP agents (Lin and Carley, this issue). The CORP agents deal with a static version of the radar-task as we have done, and the DYCORP agents deal with a dynamic version of the same task. In creating the CORP and DYCORP agents, it was necessary to create models of boundedly rational agents. Within CORP, for example, there are multiple agent models: the experiential agent, the probabilistic experiential agent, and the procedural agent. Of these agent models, the least sophisticated is the experiential agent. The experiential agent is information limited and effectively over confident in its decisions, but acts in a deterministic fashion given its information as though it has perfect recall. The probabilistic experiential agent is more sophisticated as it acts in a stochastic fashion given its information as though it has imperfect recall much in the way that Soar does. Increasing the sophistication of the agent tends to make it more Soar like. The CORP/DYCORP agents, unlike the Soar agents, can only do choice tasks. Whereas, Soar's architecture can be used with a wide range of tasks. Both CORP/DYCORP agents, unlike the Soar agents, can only do choice tasks. Whereas, Soar's architecture can be used with a wide range of tasks. Both CORP/DYCORP agents and Soar agents are complex adaptive agents. Unlike a Soar hierarchy, a CORP hierarchy is similar in form to a parallel distributed processing system. CORP agents operate in a numerical environment; whereas, Soar agents operate in a symbolic environment. CORP/DYCORP agents are based on standard learning theory, and learn exclusively through feedback. In contrast, Soar agents learn through experience and the chunking procedure. Soar embodies

[3]When we turned the learning switch on, we found that Soar tried to learn from degree to which the previous model matched, rather than from the features of the model (or current situation) that matched. This behavior is not desirable. In order to let Radar-Soar learn from model features, the code has to be rewritten significantly. An effort in this direction is the code written by Papageorgiou (1992).

[4]Papageorgiou (1992) shows that if Soar agent makes decisions based on guessing when it faces a case which the agent has never seen, the performance of 52.5%, see Carley (1993). This means Soar's decision making simply by guessing cannot fully capture the Humans' behavior. It also suggests that the main mechanism for learning in Soar, learning through chunking, is insufficient to capture social behavior and organizational learning that occurs in the face of novel situations.

TABLE 8
Required Capabilities of Social Agent

Capability	Presence in Radar-Soar
Perception and Action	
Perceives the environment	somewhat
Physically manipulates objects	no
Moves self to different locations	no
Memory	
Location	no
People	yes—managers, no—analyst's
Task	yes
Instruction	
Can be incomplete	yes
Task Analysis	
Decomposes task	yes
Coordinates subtasks for self to do	yes
Communication Skills	
Asks questions/Provides answers	yes, but limited
Gives commands/Receives commands	yes—manager, but limited
Talks to a single individual/Talks to a group	to individuals
Social Analysis	
Models of other agents	yes
Model of organization	no

a well tested model of human cognition and is consistent with much of what is known about human cognition; whereas, CORP/DYCORP embody only simple incremental learning theory and are consistent with fewer finding on human cognition. Unlike Soar simulations, CORP/DYCORP simulations run so quickly that Monte Carlo analysis is feasible. Future work should consider what relative knowledge of social behavior is gained by moving from organizations of simple adaptive agents to organizations of Soar agents.

CONCLUSION

Carley et al. (1991) define the base set of characteristics or abilities that an artificial agent would have to possess and express to be social (see Table 8). It should be noted that humans are presumed to have all these capabilities. By examining Radar-Soar on this scale we can see how much more needs to be added to the basic Radar-Soar agent to get agents that exhibit social characteristics.

Radar-Soar, e.g., has limited abilities to do task and social analysis. The task the Radar-Soar agents are engaged in is sufficiently complex that greater task analysis is warranted and some interaction among agents is expected. This research suggests that to perform a task with a certain complexity, it is necessary for the agent to have certain social capabilities.

Systems such as Radar-Soar allow us to examine the complex interrelationships among social behavior and task behavior. Even this simple model enables the researcher to locate gaps in our understanding of how organizations work (e.g., by seeing the strong relationship between feedback and organizational design). By locating such gaps we can advance our understanding of organizational behavior.

REFERENCES

Aldrich, H. E. (1979) *Organizations and Environment*, Englewood Cliffs, NJ: Prentice Hall.

Arrow, K. J. and Radner, R. (1979) Allocation of resources in large teams. *Econometrica* **47**: 361–385.

Carley, K. M. (1991) Designing organizational structures to cope with communication breakdowns. *Industrial Crisis Quarterly* **5** (1): 19–57.

Carley, K. M. (1992) Organizational learning and personnel turnover. *Organization Science* **3** (1): 2–46.

Carley, K., Kjaer-Hansen, J., Newell, A. and Prietula, M. (1992) Plural-Soar: A prolegomenon to artificial agents and organizational behavior, in Masuch, M. and Warglien, M. (eds.) *Artificial Intelligence in Organization and Management Theory*, Amsterdam, The Netherlands: Elsevier Science Publications, 87–118.

Carley, K. M. and Lin, Z. (1995) Organizational designs suited to high performance under stress. *IEEE—Systems Man and Cybernetics* **25** (1).

Carley, K. M. and Lin, Z. (1993) Maydays and Murphies: A study of the effect of organizational design, task, and stress on organizational performance, working paper, CMU-SDS.

Carley, K. M. and Newell, A. (1990) On the nature of the social agent. Presented at the American Sociological Association Annual Meeting, Washington, DC.

Carley, K. M. and Prietula, M. (1994) ACTS theory: Extending the model of bounded rationality, in Carley and Prietula (eds.) *Computational Organization Theory*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Cyert, R. and March, J. G. (1963) *A Behavioral Theory of the Firm*, Englewood Cliffs, NJ: Prentice-Hall.

Durfee, E. H. (1988) *Coordination of Distributed Problem Solvers*, Boston, MA: Kluwer Academic Publishers.

Durfee, E. H. and Lesser, V. (1988) Predictability versus responsiveness: Coordinating problem solvers in dynamic domains. *Proceedings of the AAAI-88 Seventh National Conference on Artificial Intelligence*, St. Paul, MN.

Feldman, M. and March, J. (1981) Information as signal and symbol. *Administrative Science Quarterly* **26**: 171–186.

Galbraith, J. (1973) *Designing Complex Organizations*, Reading, MA: Addison-Wesley Publishing Company, Inc.

Hollenbeck, J. R., Sego, D. J., Ilgen, D. R. and Major, D. A. (1991) *Team Interactive Decision Making Exercise for Teams Incorporating Distributed Expertise (*TIDE^2*): A Program and Paradigm for Team Research*, Tech. Rep. No. 91-1, East Lansing: Michigan State University, Department of Management and Psychology.

Laird, J., Newell, A. and Rosenbloom, P. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence* **33**: 1–64.

Laird, J. E., Rosenbloom, P. S. and Newell, A. (1986a) Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning* **1** (1): 11–46.

Laird, J. E., Rosenbloom, P. S. and Newell, A. (1986b) *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*, Boston, MA: Kluwer Academic Publishers.

Lindblom, C. E. (1959) The science of muddling through. *Public Administration Review* **19**: 79–88.

Lin, Z. and Carley, K. (this issue) DYCORP: A computational framework for examining organizational performance under dynamic conditions. *Journal of Mathematical Sociology*.

March, J. G. and Simon, H. A. (1958) *Decision-Making Theory Organizations*, New York: Wiley.

Newell, A. (1990) *Unified Theories of Cognition*, Cambridge, MA: Harvard University Press.

Newell, A., Yost, G., Laird, Rosenbloom, P. S. and Altmann, E. (1991) Formulating the problem-space computational model. CMU Computer Science, A 25th Anniversary Commemorative.

Papageorgiou, C. P. (1992) Cognitive model of decision making: Chunking and the radar detection task. *CMU-CS Bachelors Thesis*, Pittsburgh, PA.

Rosenbloom, P. S., Laird, J. E. and Newell, A. (1988) The chunking of skill and knowledge, in Bouma, H. and Elsendoorn, A. G. (eds.) *Working Models of Human Perception*, London, England: Academic Press, 391–410.

Scott, W. R. (1987) *Organizations: Rational, Natural, and Open Systems*, Second Edition, Englewood Cliffs, NJ: Prentice-Hall, Inc.

Simon, H. A. (1973) Applying information technology to organizational design. *Public Administrative Review* **33**: 268–278.

Steinbruner, J. D. (1974) *The Cybernetic Theory of Decision Process*, Princeton, NJ: Princeton University Press.

Thompson, J. (1967) *Organizations in Action*, New York, NY: McGraw Hill.

Wilensky, H. (1967) *Organizational Intelligence: Knowledge and Policy in Government and Industry*, New York: Free Press.