Decision Support Systems 8 (1992) DSS00304 S
North-Holland

Distributed artificial intelligence for group decision support

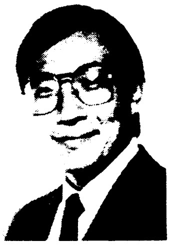
Integration of problem solving, coordination, and learning

Michael J. Shaw
University of Illinois at Urbana-Champaign, Urbana IL, USA

Mark S. Fox
University of Toronto, Toronto Ont., Canada

Decision support systems (DSS's) for aiding group problem-solving situations have become increasingly important for supporting and coordinating complex organizations. This paper describes a framework for designing, developing, and formalizing group problem-solving systems based on distributed artificial intelligence (DAI). Among the issues, we find the coordination mechanisms and the learning schemes to be of particular importance in supporting group problem solving. Two implementation examples, one on a network of expert systems, one on a multi-agent concurrent design system, are used to illustrate the distributed artificial intelligence approach to group decision support.

Keywords: Group decision support; Distributed artificial intelligence; Multi-agent learning



Michael J. Shaw is an Associate Professor of Information Systems at the Department of Business Administration, University of Illinois at Urbana-Champaign. He has also been a research faculty member at the Beckman Institute for Advanced Science and Technology, a research center on the same campus for the study of intelligence, where he is currently coordinating the research program in *decision support systems*. His research interests include intelligent scheduling, machine learning, distributed artificial intelligence, group decision support, and their applications to computer integrated manufacturing and financial management.
Correspondence to: M.J. Shaw, Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.

1. Introduction

The technological breakthroughs of the last decade have made computers an essential contributing element to the decision-making process in organizations. Nowadays, computers, with the ever-growing computational, problem-solving, and communications capabilities, are increasingly being used to provide decision support to individual decision makers, or they themselves function as decision units utilizing artificial intelligence. This new era of information technology has brought along a number of challenging problems. In particular, communications networks are emerging as a medium for coordinating organizational ac-

Mark S. Fox received his B.Sc. in Computer Science from the University of Toronto in 1975 and his PhD in Computer Science from Carnegie Mellon University in 1983. In 1979 he joined the Robotics Institute of Carnegie Mellon University as a Research Scientist. In 1980 he started and Was appointed Director of the Intelligent Systems Laboratory. He co-founded Carnegie Group Inc. in 1984, a software company which specializes in knowledge-based systems for solving engineering and manufacturing problems. Carnegie Mellon University appointed him Associate Professor of Computer Science and Robotics in 1987. In 1988 he was appointed Director of the new Center for Integrated Manufacturing Decision Systems, which is one of the largest centers in the U.S. for research in intelligent systems to solve engineering and manufacturing problems. In 1991, Dr. Fox received the NSERC Research Chair in Enterprise Integration and was appointed Professor of Industrial Engineering, Computer Science and Management Science at the University of Toronto. Dr. Fox pioneered the application of Artificial Intelligence to factory planning and scheduling problems, project management, and material design. He was the designer of PDS/GENAID, a steam turbine generator diagnostic system, which was a recipient of the IR100, and was the creator of SRL from which Knowledge Craft, a commercial knowledge engineering tool, was derived. Research interests include enterprise Integration, constraint directed reasoning and applications of artificial intelligence to engineering and manufacturing problems. Dr. Fox has published over 50 papers, and is a Fellow of AAAI, and CIAR/PRECARN, a AAAI councilor, and a member of ACM, IEEE, SME, CSCSI and TIMS.

tivities in an inherently distributed environment. These advances necessitate the development of some new principles of information system design that consider the distributed, coordinated nature of group problem solving.

We can use a common problem faced by the manufacturing industry today to illustrate this need. It has become increasingly important, due largely to global competition, to achieve shortened cycle times for developing products to meet market demands. As a result, there is a great need to coordinate the decision processes among design engineers, manufacturing engineers, schedulers, marketing managers, financial officers, etc., in an organization. Without proper coordination in the product designs, production schedules, financial analysis, and marketing plans made by the various departments, there would be conflicts among these decision processes and the resulting modifications needed would substantially lengthen the product development life-cycle. The advances in information technology can enable computer-supported coordination to support group problem-solving processes within organizations [4;36;37;20;22]. In this paper, we shall describe a distributed artificial intelligence (DAI) framework for facilitating such coordination in supporting group problem solving activities.

A number of research areas have emerged recently that concern the issues related to information systems for supporting group problem solving. Several different names have been used to describe this type of systems, such as distributed problem solving systems [15], computer-based systems for cooperative work [29], group decision support systems [9;42;43], electronic meeting systems [16], electronic brainstorming systems [57], groupware [19], distributed reasoning systems [2], and cooperating knowledge-based systems [14]), among others.

A general DAI system consists of a group of problem-solving agents collaborating in finding the solutions to given problems. In this paper we will identify the design considerations for such DAI systems and the possible mechanisms for coordinating the problem-solving activities. By developing a DAI framework for supporting group problem solving, the DSS component is taking a more active role in guiding the decision processes. Such a DAI approach can support a group of decision makers either simultaneously, such as the case in a electronic meeting system [16], or it can support asynchronous group problem-solving activities, such as the case in office information systems [61] or concurrent engineering design systems [22;8]. We use the term group problem-solving systems to encompass these different variations, and the information system for supporting group problem solving is referred to as a group decision support system (GDSS). This paper will discuss the important features, the design issues, and the necessary functions that have to be incorporated in a DAI system to effectively support group problem solving.

DAI may fill an important gap of the existing research in the DSS literature, especially in the areas related to developing and analyzing GDSSs. Nunamaker et al. [42] articulated three factors important for developing GDSSs: user profile, task domain, and technology. The technology used by the existing GDSSs has been mostly focused on facilitating the group communication, structuring the decision processes, integrating the decision-support software tools, and guiding the group interactions. DAI can naturally provide all these capabilities. In addition, just as knowledge-based expert systems can demonstrate expert-level performance by incorporating human judgement and heuristics, so can a DAI system demonstrate group intelligence by incorporating the effective knowledge and strategies for group problem-solving. Essentially, the information system incorporating DAI serves as an invisible manager supporting the group of problem-solving agents (i.e., human decision makers or AI nodes), overseeing the group problem-solving process, and making sure that the problem is solved by the group in the most effective fashion.

A critical component in developing DAI-based system for group problem solving is the coordination strategies used by the agents: the group meta-level knowledge for enticing the agents to work with each other effectively. Similar to the way expert systems are inspired by the heuristic problem-solving process used by individual human experts, the effort of developing DAI system can benefit from studying how a group of people work together in collaborative work. The challenge is for each individual problem-solving unit (synonymously, agent) to coordinate, in a most efficient and goal optimizing way, its interactions with those of other agents. Coordination can be exerted in the form of passing data, goals, preference, partial solutions/plans, or constraints.

Furthermore, this paper is to develop multi-agent learning schemes for enhancing the performance of DAI systems. It has become commonly recognized in the AI community that learning and problem-solving should be well integrated in an intelligent system. In single-agent problem-solving systems, learning capabilities are indispensable for knowledge acquisition and refinement. Learning capabilities are even more important in multi-agent problem solving due to the constant interactions and information-sharing among agents. Incorporating the right learning schemes could improve the problem-solving performance of DAI systems. Furthermore, we will show a multi-agent learning approach which use the DAI framework with a distributed problem-solving strategy to improve the performance of the learning processes.

After reviewing the research issues related to cooperative problem solving and developing a framework for characterizing such systems in section 2, section 3 focuses on the coordination mechanisms in DAI systems, which affect the problem-solving strategies and learning schemes used. In section 4, the possible schemes for carrying out learning in multi-agent problem solving are discussed, including a multi-agent induction procedure, called the distributed learning system (DLS), are discussed. Section 5 illustrates two exemplary GDSS systems incorporating DAI: the Networked Expert System Testbed (NEST) – for group problem solving, and the Design Fusion system for facilitating concurrent design.

2. Distributed artificial intelligence and DSS research

2.1. Group decision support

The demand for GDSSs arises due to two conflicting organizational situations: the requirement for more information sharing in organizations as a means to cope with the environmental complexity ad uncertainty, and the desire to facilitate coordination among a group of decision-making agents [17]. The purpose is to increase the effectiveness of decision groups by facilitating the interactive sharing and use of information among group members and also between the group and the computer.

A group decision support system improves the process of group decision making through the use of communication, computing and decision analysis techniques. By effectively coordinating the exchange of information, it provides new approaches for organizational decision making and thus facilitates the solving of unstructured problems by a group. DeSanctis and Gallupe [17] outline three levels of GDSSs. Each level alters the group communication process to a different degree and requires more sophisticated technology. Level 1 only facilitates information exchange by removing common barriers (i.e. electronic messages, anonymous input), so there is a small degree of communication change. Level 2 makes qualitative modeling tools available to the entire group, thus reducing uncertainty in the decision making process. Level 2 systems incorporate a greater degree of communication change than Level 1. Automated group structuring techniques, such as the Delphi method and Nominal Group Technique, can be applied at Level 2. A Level 3 GDSS controls the communication process by imposing communication patterns on the group. Rules are developed to control the pattern, timing and content of the information exchange. The rules, which are applied to the group meeting procedure, can be pre-set during development or selected by the group. The advancement in developing Level 3 GDSS is very much needed for truly achieving group decision support (Benbasat and Konsynski [1988]), and the DAI framework described in this paper is aimed at supporting group problem-solving functions as characterized by Level 3 GDSSs.

2.2. A taxonomy of group problem solving systems

The group problem solving system discussed in this paper is a DAI system consisting of a network of intelligent problem-solving nodes. An example of such a system is a network of rule-based DSSs, each of which incorporating an area of expertise as represented by the content of its rules. There are three different types of group problem-solving systems that have been developed:

Type 1: Collaborative reasoning systems. The agents in this type of systems would be solving the same problem collaboratively. The main issue here is not the decomposition into subproblems assigned to the agents; rather, the focus is usually put on guiding and coordinating the interactions among the participating agents, so that the problem can be solved jointly by the group simultaneously. The PLEXYS system [42;43]), COLAB [57], and the concurrent design system described in [8;20;22] all fall into this category. The collaboration among the agents is enforced through information sharing, mutual stimulation of ideas, and helping the group focus on the most relevant information for reaching the solutions.

Type 2: Distributed problem-solving systems. In these systems, the overall problem to be solved is decomposed into sub-problems assigned to the agents, each agent, asynchronously, would plan its own actions and turn in its solutions to be synthesized with the solutions of other agents. The agents in these systems use either *task sharing* [55;56] or *data sharing* (e.g., [32]) to cooperate with other agents. The office information system described in [61], Design Fusion system described in [20], and the scheduling system describe in [50] are examples of these systems. Because the agents can work on different parts of the problem in an asynchronous fashion, distributed problem-solving systems enjoy more flexibility. On the other hand, the properly designed coordination mechanism becomes especially important for these systems.

Type 3: Connectionist systems. The third type of multi-agent systems use agents as the basic computational elements. These agents individually are just simple computing units and are not intelligent; but together they can solve complicated problems quickly. A typical example is the society-of-mind model described in [39]. Unlike the previous two types of systems, where the agents are intelligent problem solvers, the agents in the connectionist model are only simple computation units. The agents learn to solve problems more effectively by adjusting their connections with each other. Such a connectionist model can help real-world systems to achieve self-organizing, adaptive, coordination between the agents. For example, [45] described a connectionist system, called CASCADE, for performing material handling in a discrete parts manufacturing environment.

2.3. *The strategic considerations: A framework for DAI*

Although the agents, the problems to be solved, and the strategies used are different in the afore-mentioned systems, it is nevertheless possible to construct a general framework to describe the strategic considerations shared among these systems. These considerations are summarized below.

(1) Goal Identification and task assignment. There are two different types of problem- solving processes that can be used by the group of agents: in the first type, the problem is presented to the whole group and the agents would collectively carry out the deliberation process, which usually consists of issues identification, proposing solutions, discussion, prioritizing, and finalizing the group solutions. For the second type of group problem solving, the tasks involved in having the problem solved are structured and known, and the common strategy used consists of four steps: problem decomposition, task assignment, local problem solving, and solution synthesis. Generally, the former approach is often used in collaborative reasoning systems; the latter is used in distributed problem-solving systems previously defined.

(2) Distribution of knowledge. In designing a distributed knowledge-based system, the set of domain knowledge possessed by the group of agents can be distributed in different fashions. In a sense, the consideration of knowledge distribution is similar to the design consideration of placing copies of data files in distributed databases - the copies of different areas of knowledge should be placed according to the utilization by each agent. There may be different degree of redundancy in the agents' knowledge bases. In one extreme, the group of agents have exactly the same knowledge; in the other extreme, each agent in the group possesses different area of knowledge, without any overlap whatsoever. For some

systems, however, the knowledge distribution is given and fixed.

(3) Organization of the agents. The organizational structure of the agents would determine the amount of information processed and the coordination necessary for the agents to operate efficiently. [35] evaluated a variety of organizational structures and compared them on three dimensions: amount of processing needed, amount of coordination required, and degree of vulnerability. [21] addressed the issue of organization structure and problem characteristics; he articulated the two opposing forces of task complexity and uncertainty, and related them to the issue of bounded rationality of the agents in an organization (or a computer system). [12], on the other hand, underscored the importance of considering the transitory nature of organizational decision making; he argued for incorporating flexibility and the dynamic performance of the agents in an organization. In the GDSS environment, the group, for example, can have a central coordinator which gives commands to the other agents in a hierarchical fashion; alternatively, the agents can have a heterarchical structure, sometimes with the flexibility to dynamically form subteams. In addition, the agents may also use a market structure, such as the case for the DAI systems based the contract-net framework [56].

(4) Coordination mechanisms. Coordination is necessary in multi-agent problem solving for resolving conflicts, allocating limited resources, reconciling different preferences, and searching in a global space for solutions based on local information. Coordination mechanisms can be based on a variety of information passed among the agents, such as data, new facts just generated, partial solutions/plans, preferences, and constraints. A number of coordination mechanisms have been developed for DAI systems, each of which with its unique protocol for determining the timing of activities, triggering of events, action sequences, and message content in the coordination process. The role of coordination in multi-agent problem solving is discussed in detail in Section 3.

(5) Learning schemes. Learning should be an integral part of problem solving for improving the strategies, knowledge, and skills employed in the process. There are several learning processes that can be incorporated in group problem solving systems. It can be in the form of data exchange, knowledge transfer, or heuristics migration, where the learning mechanisms involved are relatively simple. It can also be done by extending machine learning techniques developed for single-agent systems, such as explanation-based learning, case-based reasoning, or inductive learning, to the multi-agent systems, where one agent can learn by observing other agents. In the organization context, the learning processes interact with the dynamic performance of the agents [12]. [34] showed how the learning effects can be affected by coordination among the agents. Of particular interests are the use of group processes, such as group induction, nominal group techniques, or brain storming, for achieving the learning effects among the whole group. These group processes use structured sessions of information exchange to develop new concepts, which would lead to solutions not attainable by any of the agents alone. They are also good examples for illustrating the complementary role played by the problem-solving and learning activities.

In implementing a group problem-solving system, these five strategic considerations can be implemented in the form of group meta-knowledge [7]. The group meta-knowledge would contain a variety of group functions and strategies for the group to conduct problem solving effectively. An example of such implementation is described in Section 5. Directly affecting how the multi-agent system operates, this group meta-knowledge can be embedded in the network through which the agents communicate. [55] described the use of a 'problem-solving layer' for incorporating such group meta-knowledge in the layered network architecture. It can also be viewed as a 'protocol' which every agent has to observe for the group to operate efficiently. Depending on the organization of the system and the knowledge distribution, group metaknowledge would be stored either in the agents' local knowledge bases or a shared memory space in the network. It is used by the DAI system to direct and coordinate the group problem-solving sessions.

3. Coordination mechanisms for group problem solving

Coordination is the key design component for group problem-solving systems. Since each problem-solving agent only possesses local view and incomplete information, it must coordinate with other agents to achieve globally coherent and efficient solutions. The design of coordination can be viewed from three different perspectives: the information content, the exercise of control, and the coordination mechanisms. Coordination can be achieved through passing different types of information among the agents, such as data, new facts just generated, partial solutions/plans, preferences, and constraints. The initiative to coordinate may result from a variety of means of control: it may be self directed, externally directed, mutually directed, or a combination of them [13].

Since coordination represents the major design components in extending problem-solving methods to the multi-agent environment, it has attracted the most attention for research in the literature. Many coordination mechanisms have been developed for DAI and group problem-solving systems, some representative ones are described below. Most of group problem solving situations, however, sometime require the use of more than one kind of coordination mechanisms.

1. Coordination by revising actions. One of the primary factors that necessitates coordination is the potential conflicts underlying the actions of the individual problem-solving agents. The objective of coordination, then, is to derive a solution or a plan of actions for the group such that all the conflicts are avoided. [10] used the collision avoidance problem in air traffic control to illustrate coordination mechanisms based on passing the information constantly among problem solvers, each of which guides the course of an aircraft. The flight plan would be revised if there is a risk of conflict with the plan of other aircrafts. [5] developed a conflict-resolution framework for coordinating the solution processes among multiple agents performing tasks related to engineering design.

2. Coordination by synchronization. In a multi-agent system, the action of an agent usually affects some other agents. The objective of coordination in a way is to regulate the interactions among agents and to control the timing and sequence of these interactions. Since the agents generate new facts throughout their problem-solving processes, these new pieces of information need to be properly coordinated so that they would not interfere with each other. In addition, the new facts generated by one agent may activate the rules in the knowledge base of another agents. [13] and [24] incorporated communications primitives, similar to the ones used in distributed operating systems, for synchronizing the problem-solving process of the agents. The newly discovered facts may also change the assumptions held by other agents; as a result, a distributed version of the true-maintenance mechanism may be needed for keeping the knowledge bases consistent [2].

3. Coordination by negotiation. Negotiation is a widely used form for mutually directed coordination: the process involves two-way communication to reach a mutually agreed course of actions. The contract-net mechanism described in [55] uses negotiation to coordinate the sharing of problem-solving tasks among agents, in which the negotiation process is done by contract bidding. [14] adopted the negotiation process to maintain the consistency of different agents' plans. [10] also used the negotiation for finalizing revision of the agents' plans. [58;59] developed a system for labor contract negotiation that adopted case-based reasoning to facilitate the process.

4. Coordination by structured group mediation. Structured group processes, such as the nominal group technique, the Delphi technique, and the brainstorming process, are a special type of coordination based on multiple rounds of group interactions. They are developed for guiding the group to reach satisfactory solutions in a systematic manner. Recent work on collaborative work [57] and group decision support [1;42;43] use these group processes to mediate group problem-solving among multiple human agents. It should be feasible, but yet to be tested, to implement these schemes for coordinating the problem-solving activities among AI agents. [25] studied the user interface in GDSSs; they described the various group processes, such as voting, brainstorming, issue analysis, and negotiation that have been supported by GDSSs.

5. Coordination by opportunistic goal satisfaction. The blackboard model for problem solving [41] has been used extensively in DAI systems [18]. The blackboard model provides a paradigm for coordinating the multiple problem-solving agents (i.e., knowledge sources), which share the blackboard, to opportunistically contribute to the group solution process. However, the use of a single blackboard in a DAI system implies shared memory, which would restrict the architecture of the DAI system. [51] described an approach incorporating a distributed version of the blackboard model but retained the opportunistic mechanism for coordinating the agents, each of which is a stand-alone expert system. [41] described two DAI systems, Cage and Poligon, that extends the blackboard model to concurrent problem solving.

6. Coordination by exchanging preferences. A group of researchers have applied the game-theoretic view to multi-agent problem solving. They focus on how the group of autonomous, self-interested AI agents should interact with each other in achieving globally satisfactory solutions. [23] developed a coordination scheme that does not need communication. The agents, however, need to know the possible actions and the corresponding utility functions of one another. Taking a game-theoretic approach, [47] described a coordination scheme between agents based on the exchange of the payoff matrices.

7. Coordination by constraint reasoning. In some group problem-solving situations, the major purpose of coordination is to find a common feasible solution space by taking into account the collective set of constraints posted on the individual agents. Since the constraints of one agent may affect the optimal decision of another agent, a mechanism to coordinate the constraint interactions becomes crucial for the group of agent to reach a solution satisfactory to all members. [48] developed a constraint-directed negotiation approach allocating resources among multiple agents. [22] describes a multi-agent design system, called Design Fusion system, that use constraint reasoning as the engine to direct the problem-solving processes among multiple agents.

4. Multi-agent learning

Machine learning is the study of how to develop computer programs that enable an intelligent system to learn. By learning we mean the system can improve its performance through acquiring new knowledge, refining existing knowledge, using better strategies, or memorizing cases previously proven to be successful. Potentially there are a number of benefits for developing machine learning techniques for multi-agent systems. First, since multi-agent systems provide parallel processing capability in performing the learning process, they can achieve substantial speed-up in learning new knowledge. Second, machine leaning capabilities can enhance the problem-solving ability of the multi-agent system and improve its performance. Third, understanding the learning processes in a multi-agent system can help develop better problem-solving strategies and coordination mechanisms. The fact that a group collectively can perform problem-solving tasks better than what the sum of the individuals' abilities can do is primarily due to the learning processes that go on in group problem solving. This phenomenon is referred to as emergent intelligence in DAI [50]. It is this emergent intelligence of DAI systems, that a group of agents collectively can offer something not available in the individuals, that makes DAI a potentially powerful problem-solving tool. The learning processes that go on among the agents may be the key factor resulting in emergent intelligence.

In a DAI system, two types of learning may occur: the agents can learn as a group, while at the same time, each agent can also learn on its own by adjusting its views and actions. As a group, learning takes effect in a DAI system in the form of (1) better coordination or (2) more efficient task and resource allocation. The improved coordination can be achieved by information sharing, knowledge sharing, or more efficient communications among the agents. Whereas the task and resource allocation process can be improved by learning the specialization (i.e., knowledge distribution) of the agents (e.g., agent x is good at performing task A), by learning the group characteristics (e.g., agents y and z work well as a team), by learning the patterns of tasks (e.g., for a given type of problem, the past experience indicated that it is easier to break the problem into two tasks, C and D, and do D first), and finally, by learning such environmental characteristics as user preferences, processor reliability, or future jobs [50].

[39] developed a model for intelligence based on the theory that a human problem-solving process consists of a set of smaller processes, each of which conducted by an agent with specialized responsibility. The problems handled by the agents in Minsky's model, referred to as the society of mind, have much smaller granularity than ones handled by the agents described in NEST. The major thrust of the society-of-mind model is to explain intelligence as a combination of simpler things executed by agents. Although none of these agents individually is intelligent, collectively they can solve complicated problems and do that quickly.

The society-of-mind model captures both the problem-solving as well as the learning processes. The key is to emphasize not only how the agents solve the sub-problems, but also the inter-relationships between the agents in the whole process. The idea is that in the course of solving some problem, certaIn agents must have stimulated certain other agents. A reward system is incorporated so that if agent A has been involved in stimulating agent B, it will be easier for A to stimulate B in the future. The group of agents activated in having a problem solved would have strong connections with each other. Such connected groups should make it easier to solve the same problem the next time by simply reactivated the same group. Minsky developed a theory of memory based on this connectionist view, in which the group of agents involved in solving a problem successfully was referred to as the knowledge line.

Learning, based on this model, is not simply the acquisition of a single concept. It involves assigning priorities (weights) to competing concepts which can be represented by a hierarchy of agents. A crucial aspect of the model is to deal with knowledge refinement in such hierarchies. Minsky argued that it is not enough to have many kinds of reasoning; one must know which to use in different circumstances - i.e., the knowledge about how best to use what was already learned. Minsky related this to the issue of the organization of the agents in the following characterization, in what he referred to as the Papert's principle: "Some of the most crucial steps in mental growth (i.e., learning) are based not simply on acquiring new skills, but on acquiring new administrative ways to use what one already knows."

In other words, in the multi-agent system, it is necessary to learn how to organize the group of agents in a problem-solving process most efficiently for solving that problem. In the society-of-mind model, as the problem-solving process help accumulate more low-level agents, it is necessary to create new levels of agents to administer the low-level agents. The group of agents thus would grow increasingly into a multi-level hierarchy. This emphasis on organizational structures in dealing with the multiple agents is consistent with our DAI framework described in section 2. Since this model uses fine-grained problems, the agents are simple computation elements. Hierarchies can provide good coordination for the group of agents with the least amount of overall communications activities required [53;21]. This type of systems can learn the best organization simply by adjusting the interconnections between units. By enabling the evolution of the system from a bad organization to a good one, the approach has the flavor of a self-organizing systems [3].

[60] described a connectionist learning scheme for organizations based on the garbage can model [12]. The basic idea is that the logic of organization intelligence is a temporal one: it emerges as a result of the confluence of partially independent flows of problems, solutions, decision makers, and choice opportunities. An interesting result of this approach is that the emergence of organizational learning phenomena can be addressed as a by-product of the choice activity in organizational decision making. This Finding is consistent with our view on emergent intelligence discussed earlier.

This ability to organize the processing agents as needs arise would be quite significant in a GDSS environment, where the efficiency and effectiveness of the group decision processes are affected by the organizational structure of the group. The weight adjustments used in the society-of-mind approach may be related to the amount of communication among agents- the agents learn to organize themselves by learn to better coordinate. Improved coordination can be achieved by getting to know such information as which is the most likely agent to have a given area of expertise or the preference of other agents in sharing tasks.

Concept learning is one of the most developed areas in machine learning, where the objective is to derive concept descriptions satisfying all the training examples [44]. Starting with the given set of training examples, the learning process employs a series of generalization and specialization steps to search through the space of all possible concept descriptions [[54]; this process continues until the concept descriptions that satisfy all the descrIptions of the training examples are found.

A direct extension of the concept-learning algorithm for single agent is the group induction process. In group induction, the agents of the system engage in collecting evidence, generating hypotheses, and evaluating hypotheses in order to discover new concepts collectively. Each agent makes its individual observations, forms hypotheses based on the observations, and keeps searching for new evidence for modifying the hypotheses. As more observations are made, evidence may confirm or disconfirm hypotheses, thus strengthening or weakening the agents' beliefs about the hypotheses. This process continues until new concepts are derived that are supported by the observations and the agents' beliefs. The crucial design in such a group induction process is the coordination mechanism incorporated for the agents to interact with each other. Proper coordination can greatly accelerate the learning process by having the agents share their beliefs as well as the newly generated hypotheses that appear to be successful.

A possible source of inspiration for understanding the group induction process comes from the psychology literature where researchers have been conducting behavioral experiments to study group induction. For example, [31] described an experiment on group induction to address the issue whether "a cooperative group is able to induce a general principle that none of the group members could have induced alone, or the group merely adopt an induction proposed by one or more of the group members. They concluded that the group were remarkably successful in recognizing and adopting a correct hypothesis if the hypothesis was proposed by one or more members. In contrast, group induction in the strong sense that a correct group hypothesis was derived that none of the group members had proposed individually was extremely rare. Their findings can be summarized by the following propositions: *Proposition 1*, Group induction is advantageous over single-agent induction primarily due to the agents' sharing of hypotheses and new facts generated, not necessary because of that the group together can generate new hypothesis not derivable by the individual members; and *Proposition 2*, In the group induction process, the sharing of hypotheses and new facts generated among the agents would result in the identification of correct group hypotheses in fewer iterations.

The first proposition attempts to identify the key element of group induction; the second proposition attempts to explain the underlying reason for the phenomenon. Together, based on the evidence presented by the experiment, these two propositions point out the importance of information sharing in a group induction process. By the same token, designing algorithms for achieving the same learning effects requires the incorporation of effective coordination mechanisms that would facilitate information sharing.

The major benefit of having the group of agents is their sharing of the newly generated hypothesis in each iteration, which Increases the probability that a correct hypothesis can be identified sooner. In a DAI system, since each AI nodes, a problem-solving agent, has a different knowledge base, the rules they use in generating new hypothesis would be different. Information sharing thus would provide greater probability to identify the correct hypothesis for concept descriptions sooner.

We can make the conjecture that group problem solving can find solution quicker because of information sharing. New facts, hypothesis, and partial solutions generated by one agent in its problem-solving are shared by other agents. These additional pieces of information shared among the agents would trigger the set of agent to conduct new problem-solving and evidence-gathering activities which would not have been performed without the agents' interactions. In turn, these triggered activities would generate new facts, hypothesis, and evidence that are shared among the agents. Thus, the amount of information shared in the agents' problem-solving efforts grows exponentially, giving the group an advantage to reach a solution much sooner. On the other hand, the exponentially growing information would mean substantially more searching efforts in screening through all the information generated. A properly designed coordination mechanism can lead the group to evaluate all the facts, hypothesis, new evidence, and partial solutions more systematically.

The primary objective of coordination, therefore, is for focusing the agents' attention on the relevant information. It also can be used to constrain the search space engaged by the group. The Delphi method is an example for such a group problem-solving method incorporating a structured coordination mechanism [33].

Since the complexity of the learning processs is due primarily to the hierarchical nature of the concepts to be learned- that is, learning has to build on intermediate results not explicitly contained in the examples- a group problem-solving approach should have the advantage of having the agents share their intermediate results, which helps facilitate the identification of the crucial intermediate concepts during the learning process and thus enables the group to reach the solutions faster.

Based on the previous discussion on group induction and distributed problem solving, a distributed learning system (DLS) was developed in pursuit of the issue of how a group of agent might conduct the concept-learning process in a cooperative fashion [52]. This algorithm basically apply distributed problem solving to the induction process just described, using the task-sharing strategy developed by Smith [55].

The DLS embodies the views underlying Propositions 1 and 2, i.e., the multi-agent version of the inductive learning process is advantageous because of the knowledge, hypotheses, and facts shared among the agents. Moreover, the solution-synthesis step also incorporates mechanisms for generating new hypotheses based on the evaluation of existing pool of hypotheses generated by the group. DLS uses the genetic algorithm to evaluate the hypotheses id to produce new hypotheses which have high success rate based on the evaluation. As shown by the empirical study described in [52], the DLS has superior learning performance comparing to the single-agent version of the same learning procedure.

The distributed learning system research helps illustrate two things: first, a multi-agent approach to learning can be advantageous over single-agent learning in terms of the learning outcome and the system's performance; second, the multi-agent learning process can be guided by the the distributed problem-solving mechanism, but special care need to be taken to ensure that the pool of hypotheses generated by the group have been fully utilized.

A hypothesis that has been central to our inquiry is that the mechanisms of multi-agent learning are not peculiar to that activity but can be subsumed as special cases of the general mechanisms for group problem solving. The electronic brainstorming process incorporated in [57] and [42], for example, is a case in point. The process is primarily aimed at structuring the group interactions for issue identification, idea (i.e., hypotheses about good solutions) generation, evaluation, and solution selection. This group problem-solving process highly resembles the group induction process discussed in the preceding section. Therefore, a group learning procedure can be used to solve problems in a DAI-based GDSS. This is especially true for problems that are relatively unstructured and fuzzy in nature, such as organizational planning [1], proposal development [57], or engineering design [5;8;20;22], where the eventual solutions are mostly satisfying rather than optimizing. The quality of the group solutions obtained in these domains depends to a great extent on whether or not sufficient considerations have been given to inducing good ideas from the agents (i.e., the hypotheses transformation step in learning) and to recognizing the information most relevant to the final solution (i.e., the credit-assignment step in learning). Such an integrated approach to problem solving and learning were pointed out in the early development of machine learning [54;30]. But this unification seems to add a new dimension to group problem solving; namely, the group interactions required in group problem solving are very similar to those in group learning. In this light, it should be quite possible to develop multi-agent learning methods for enhancing group decision support.

When a GDSS is used for supporting group problem solving and the participating agents are performing disparate tasks involved, learning processes can be used to achieve better coordination and task/resource allocation [50]. More research is needed to establish the relationships between coordination and learning in this context. [11] develops a model to formalize the interrelationships among reputation, learning, and bidding as important elements of coordination. The model explicitly recognizes that coordination can be influenced by both current and past decision processes. The model stresses opportunity for coordination improvement via reputation adjustments that constitute a kind of organization learning. It suggests a mechanism, based on the concept of bidding, whereby an organization's experiences cause each entity's reputations to eventually converge towards its true abilities, with a likely result of improved coordination in ad across decision processes. Based on similar designs, it is possible that a "learning through collaboration" mechanism Ca be developed for GDSSs. In this paper, we describe a more general mechanism in which multi-agent learning is achieved through task sharing. Since each agent is treated equally and remains anonymous, reputation is not considered.

In addition, machine learning methods may be incorporated to enhance group decision support by equipping the agents to learn the other agents' belief, preference, and the previous group solutions or cases that were developed for handling certain problems. For example, [58;59] incorporated the case-based reasoning capability in a group negotiation system called PERSUADER. In mediating the conflicts between agents, PERSUADER uses a high-level structure called generalized episode [49] to organize similar concepts in memory. The group solutions (i.e., tradeoffs and compromises that were made) proven to be successful previously for similar problems and agents involved will be stored for future retrieval for similar cases. Failed solutions will also be stored to warn the agents of potential difficulties in future similar situations. Other machine learning methods such as the explanation-based learning technique can be used to achieve the same effect. These methods enable the individual agents to learn from the interactions with the other agents in the group problem-solving situations, such as the concurrent design process in the Design Fusion system.

Based on this DAI framework, the agents can learn as a group whereas each agent can also learn on its own by adjusting its views and actions. It points to the need for more coordination for accomplishing these learning functions to facilitate group decision support. In terms of the

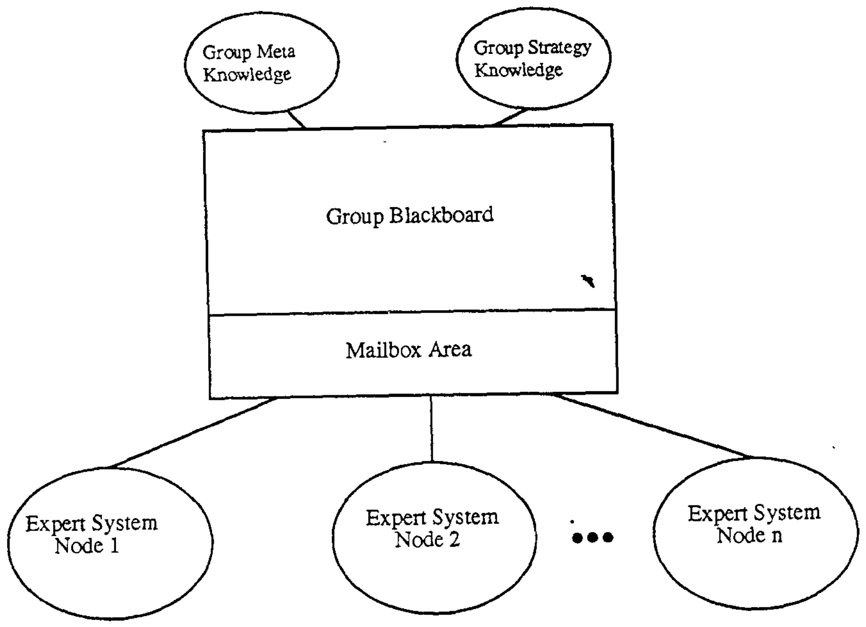


Fig. 1. The architecture of NEST.

NEST architecture, the group learning processes would refine the group meta-knowledge; the individual agents' learning processes would affect the agents' local knowledge bases. Together these learning processes would integrate with the group problem-solving processes supported by the GDSS and produce better group solutions.

5. Examples of DAI systems for group decision support

5.1. NEST: A networked expert systems testbed

A prototype version of our group problem-solving model, based on the DAI framework discussed, was implemented in an experimental network. The system, referred to as the networked expert systems testbed, or NEST, consists of a network of four expert systems. The architecture of NEST is based on a variation of the blackboard system, with "mailbox" areas added to the blackboard shared area for coordinating the agents. The architecture of NEST is shown in Figure 1.

NEST was developed using Texas Instrument Personal Consultant Plus Version 4.0, an expert system development program. Personal Consultant On-line, a supplementary tool for real-time applications, was also utilized. The prototype was installed on an IBM Token Ring Network of IBM-AT computers. PC Plus and On-line were installed locally at four nodes of the network. A distributed version of dBase III + was installed as a shared memory so all nodes could have access to it. dBase was selected because PC Plus has internal interface with it.

5.1.1. Group meta-knowledge

NEST is a generic DAI system. It was not dependent on a specific application. The group meta-knowledge of the system directs the group problem-solving processes. It asks the user for the problem description and characteristics of the problem; It also determines some group characteristics and selects the appropriate group strategy based on this information.

Corresponding to the knowledge distribution, the user describes the group problem with the aid of an expertise database. This contains a list of the available expertise on the network, indexed by a standardized classification scheme. It also indicates the node where the particular expertise is located. The database needs to be updated by a human system administrator whenever a new expert system is added to the network.

For the goal-identification and the task-assignment function, if the user is unable to solve the problem with the given expertise, the problems may be decomposed into sub-problems. The group knowledge base will select a strategy for each of the sub-problems individually. The user can then combines the results from each node and arrives at the overall problem solution. This is the task-sharing strategy for distributed problem solving.

The expertise database is also used for task assignment. Given the problem description, the expertise database is consulted to find out which nodes are capable of solving this problem. This is especially helpful in the collaboration strategy. When one node needs help and asks the strategy coordinator for assistance, the coordinator checks in the expertise database to see which nodes can provide the required expertise.

5.1.2. Coordination mechanisms

NEST uses a modification of the blackboard architecture, in which the agents, i.e., the expert systems, can communicate through two means - the blackboard or the mailbox – for coordination.

Blackboard

The blackboard communication area is implemented with the following three databases:

(1) Coordination database. This serves as a mailbox for the coordinator to coordinate agents. It is part of the blackboard and all of the local nodes have access to it. The fields of the database indicate which node sent the message, a command for the coordinator, and a message. For example, the coordinator can receive two types of messages in this database: (1) *x-node* Request *parameter-name* and (2) *x-node* Respond *status-value*.

(2) Value database. Serving as the world model for the multi-agent environment in NEST, this is a shared area where parameter values for the current problem are stored. It contains the parameter name, the value, and a timestamp. Every time a node ascertains a new value for a parameter, it updates the entry in the value database. Thus, the most recent value of the parameter is

always available. Users of the value database (i.e. other nodes) can calculate how current the value is by checking the timestamp.

(3) Status database. This database marks which nodes are currently busy so they will not be assigned a new task. A timestamp is included for start and finish of the task.

Mailbox

The second means of communications between the agents are through the mailbox area. Each node in the network has a unique mailbox database that serves as its mailbox (refer to Figure 1). Messages to the nodes from the coordinator are placed in this database. The message consists of a field indicating the type of message ("provide"), and a field containing the parameter name. The coordinator would initiate a functional expert system by sending it the notification message. The mailbox database is periodically checked by each node to see if it contains any new messages, just like the way one would check his mailbox.

State transitions in the expert system nodes

There are three possible states that an expert system node can be in. In *idle state*, the node is waiting to be initiated by a strategy coordinator or by a human user. The node periodically checks its mailbox database to see if it has any "provide" messages. If so, then it enters the *working state.*

The node is actively running rule-based consultation during the working state. When information is needed, it checks the value database for the required parameter value. If the value is not there, or if it is too old, then the functional expert system sends a "request" message to the collaboration coordinator (strategy database) to ask for help. It then enters the active *waiting state.* It returns to the working state when the parameter has a new value and the consultation continues until complete. At that time, the node will update the value database with new parameter values,

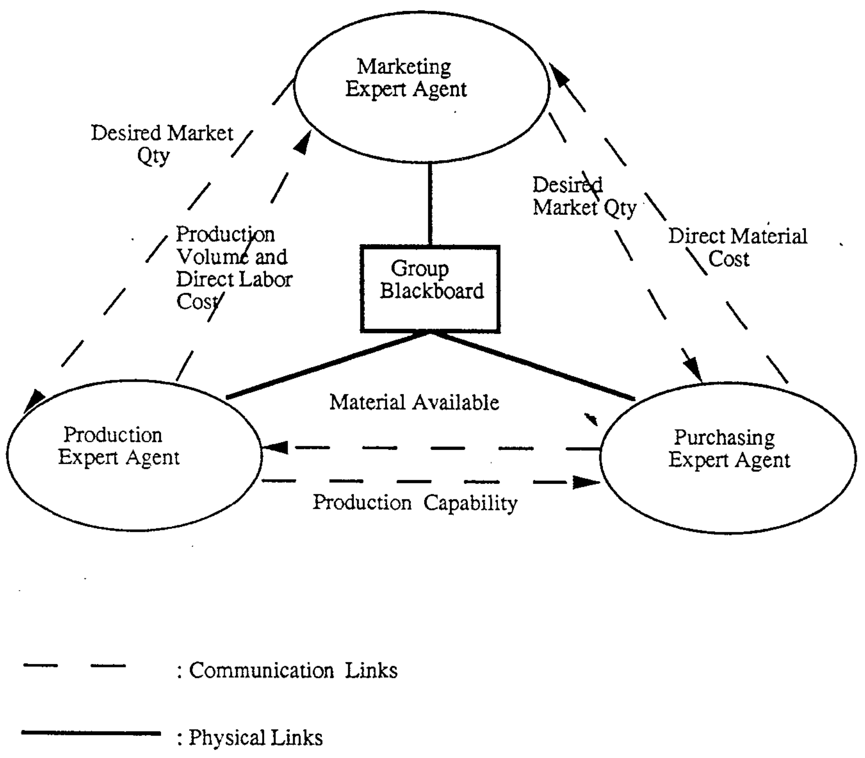


Fig. 2. A group problem solving example with NEST.

and send a "respond" message to the strategy coordinator. The node re-enters the idle state.

In the active waiting state, the node periodically checks the value database until the requested parameter value is updated. If possible, the node will continue to run the consultation, determining other parameter values and goals that do not rely upon the missing information.

5.1.3. Organization of the expert systems

The organization incorperated in NEST is a hierarchical structure, in which an expert system serves as the coordinator in the group problem-solving process. Given the initial problem description, the collaboration coordinator assigns the problem to a functional expert system node. It uses the expertise database to check the knowledge distribution and see which nodes are capable of solving this problem, and checks the status database for their availability. When the assignment is made, the functional expert system is initiated by sending it a "provide" message.

The coordinator periodically checks in the strategy database for messages from the functional expert system. In the collaboration strategy, two messages are possible. A local expert system will send a "request" message when it needs a parameter value. The coordinator determines which nodes can supply this parameter value, assigns the task to one of them, and updates the status database to reflect the assignment.

A "respond" message can also be sent to the strategy database. This indicates that a node has completed its processing. The coordinator updates the finnish time in the status database. If all nodes in the status database are done, then the collaboration process has solved the group problem.

5.1.4. A distributed problem-solving example

To demonstrate the working of NEST, in particular the collaboration strategy, a common organizational decision-making situation was used as a cooperative problem-solving example [51]. The knowledge distribution is as follows: Three functional areas – marketing, production, and purchasing – need to collaborate to reach a decision: What quantity of a product should be sent to the market? Perhaps this decision could influence the amount of advertising and sales effort.

The marketing expert initially sets a desired market quantity, based on given supply and demand information. Before it can make the final decision, Market Quantity, it needs input from the other two experts: Production Quantity and Direct Labor Cost from Production; Direct Materials Cost from Purchasing (Figure 2).

The purchasing expert receives the request for information from marketing, along with marketing's desired market quantity. Using its production constraints and other scheduling requirements, it determines how much it is capable of producing for marketing (Production Quantity). However, this quantity is also dependent on the availability of materials from the purchasing department. If the materials are not available, then the quantity will be adjusted accordingly before notifying marketing.

The purchasing expert determines if it has enough materials in inventory, or if it has to place order for production to manufacture its recommended quantity. It also calculates the direct materials cost for marketing. This decision process continues until marketing reaches a final decision. The market quantity will be adjusted in response to production's capabilities and the cost to produce. For instance, if the costs are too high, then the quantity may be increased to achieve economies of scale.

The design details of NEST and the trace of rules used in the group problem-solving session for solving this example are described in [51].

The implementation of NEST demonstrates the viability of applying the multi-agent problem-solving framework described in sections 2 and 3. It is especially interesting to be able to integrate two widely used softwares to implement the expert system nodes and the distributed databases, incorporating a variation of the blackboard model. More work would be needed to make the group meta-knowledge more flexible in dealing with different types of problems, group strategies, learning schemes, organization structures, and knowledge distribution.

5.2. A GDSS for concurrent design: The design fusion system

Mechanical designs are often composed of highly-integrated, tightly-coupled components where the interactions are essential to the execution of the design. Therefore, concurrent rather than sequential consideration of requirements, such as structural, assembly, manufacturing, and maintenance constraints, will result in superior design. Such an approach to engineering design, which simultaneously take into account several stages of design requirements in the development life cycle, is referred to as concurrent design. It has become an increasingly important manufacturing technology because by adopting concurrent design in developing new products, a company would be able to shorten the time it takes to meet market demands, thus making its products more competitive. Since the key to successfully implementing concurrent design is to coordinate the various areas of design considerations, which can be viewed as group problem solving, the process can be aided by a GDSS.

Such a GDSS is for creating an environment that surrounds the designer with experts and advisors – i.e., agents – that provides continuous feedback based on incremental analysis of the design as it evolves. These agents can generate comments on the design (e.g., manufacturability or ease of assembly), properties underlying a specific design (e.g., stresses), and portions of the geometry (e.g., the shape of a component). More than a set of software tools, the GDSS consists of a group of advisors who interact with one another and with the designer.

The Design Fusion system described in [22] is an example of such a GDSS, aided by distributed AI. The goal is to "... infuse knowledge of downstream activities into the design process so that designs can be generated rapidly and correctly." The design space is viewed as a multi-di-

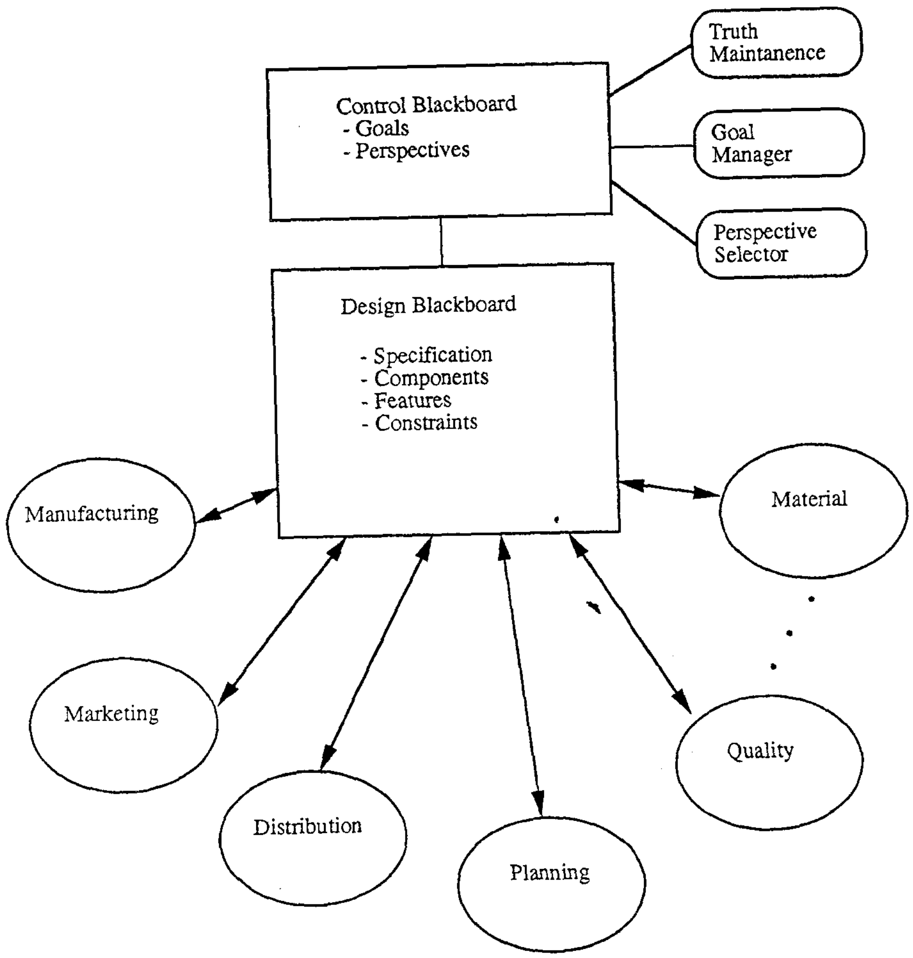


Fig. 3. Design fusion architecture.

mensional space in which each dimension is a different life-cycle objectives such as fabrication, testing, serviceability, reliability, etc. These dimensions are called perspectives, or design agents, because each dimension can be thought of as a different way of looking at the design. Figure 3 depicts the organization of the system. The designer ad three life-cycle perspectives.are represented as manufacturing, structures, and dynamics. Each of these perspectives encapsulates expertise for the relevant specialty. The perspectives monitor the design through the blackboard, which represents the design in terms of geometry, constraints, and a database of qualitative design features and quantitative parameter values derivable from the geometry.

[22] describes the problem-solving architecture for design fusion. The system can be viewed from the standpoint of the five design dimensions discussed in section 2.3.

(1) Goal identification and task assignment
Design is a search process that explores the elaboration of portions of a design resulting in a top-down decomposition of the design problem into smaller, more manageable subproblems. Subproblems are further decomposed until solvable problems are found. At any particular time during the design process, the solving of a subproblem is the goal of the designer, with the ultimate goal being the creation of an artifact or description of an artifact. To satisfy goals, the design agents make changes to the design. The design progresses through multiple levels of abstraction and refinement until all goals are satisfied. The architecture of the design system would support the process of goal formation and decomposition, and the selection of subtasks upon which to focus design attention.

The task assignment of the design system is executed in an opportunistic fashion, always selecting the perspective of a design agent that is the most relevant in terms of constraint satisfaction. The task-assignment decision is supported by the blackboard architecture.

(2) Knowledge distribution
The design space is viewed as a multi-dimensional space in which each dimension is a different life-cycle objective such as fabrication, testing, serviceability, reliability, etc. These dimensions, which form the knowledge bases of the design agents, are called perspectives. Using perspectives allows the Design Fusion system to partition the design knowledge into manageable chunks, while allowing us the flexibility to add new information to the representation.

(3) Organization structure
The design fusion system adopts a heterarchical organizational structure and control, which permits competition between agents that are cooperating toward a common goal. Heterarchical systems have many disjoint agents available for particular tasks. Coordination is by negotiation and contract based upon the marginal cost of the task [58]. Each agent in the heterarchy pursues its own goals in correspondence to the needs of others. In the design fusion system, the heterarchy provides a structure for problem-solving with multiple perspectives. Negotiations between multiple competing perspectives decide the particular design tasks to pursue in the design process.

(4) Coordination
The coordination of the design system is directed by constraints. During problem solving, design agents must cope with interactions between design constraints: the specifications, the goals of the design, and the design decisions made during the design process. The major purpose of the coordination mechanism is to provide support for the management of these interactions. It should identify which subtasks interact and what impact the design decisions will have.

Since the parameters specified by one agent may violate the constraints from other agents' perspectives, the ability to reason about the constraints of various perspectives, resolving the conflicting constraints, and reaching a feasible design by iterative adjustment would be necessary. In the Design Fusion system, the following mechanisms are used to take into account the effects of constraints: (1) Parameter revision – when a constraint is violated, the parameters involved in that constraint would have to be modified; (2) Postponement – sometime the resolution of a violated constraint is postponed to be considered at a later time, so that more critical constraints can be considered first, or that the system anticipates the violated constraint may become feasible again; (3) Revelation – resolving the conflicts between two perspectives may require the agents to reveal more detailed level of abstraction in the constraint space, so as to facilitate the reaching of a compromised solution: (4) Negotiation – agents participating in the design process needs to use negotiation when they make conflicting recommendation, or when a design decision made by an agent adversely affect the decision optimality of another agent.

The blackboard architecture enforces a shared representation of the design, which supports inter-agent communication.

(5) Learning

The very term design fusion points to the need of achieving some sort of synergistic effects among the agents in the design process. The benefit of having a group of agents in design problem solving is that everyone can contribute opportunistically and share their Thus, inherently, the Design Fusion system has adopted an integrated framework for problem solving and learning, in which learning is achieved by having the agents share their knowledge, experience, heuristics, and data.

Stronger types of learning can be achieved through the implementation of explicit learning mechanisms. For examples, since most designs bear some similarities to previous designs, it is often the case that parts of a design can be solved by remembering past experience – that is, the design process is a combination of problem-solving steps and steps that directly apply cases learned from experience [40]. In the Design Fusion, since the knowledge base is distributed among the set of design agents, these different agents should have their individual case bases to facilitate the design process. Further more, the case base should include cases about conflict resolution learned from prior experience of solving similar design problems [59], so that the coordination of multiple design agents' perspectives can be facilitated.

6. Conclusion

In this paper we have constructed a DAI framework for group problem solving, in the context of developing GDSSs. We have shown the importance of integrating problem solving, coordination, and learning in such systems. The DAI framework was illustrated by the implementations of a network of expert systems, called NEST, and the Desing Fusion system - both are multi-agent problem-solving systems based on the blackboard architecture. We also identified multi-agent learning as an important component of DAI systems. A multi-agent induction algorithm, called DLS, was developed and illustrated.

The study of NEST, Design Fusion system, and the analysis of DLS all reveal the efficacy of the DAI approach to support group problem-solving and learning tasks. The DAI approach for supporting group problem solving has several advantages: information sharing achieved by coordinating the agents; better solution quality due to agents' specialization; facilitated solution procedure by having the agents pursuing diversified path in the search space; and the improved response-time because of the parallelism among the agents' problem-solving activities. This framework can be a useful tool for developing, for example, Level 3 GDSSs outlined in [17].

Acknowledgements

We would like to thank the two anonymous reviewers, whose comments helped improve some of the sections. The example on design fusion is based on the research undergoing in the Robotics Institute of Carnegie Mellon University, as described in Finger et al. [20] and Fox et al. [22]. The able programming efforts of Barbara Harrow and Sherri Herman in developing the NEST prototype, and of Riyaz Sikora for developing DLS are also much appreciated.

References

[1] L. Applegate, T.C. Chen, B.R. Konsynski and J. Nunamaker, Knowledge Management in Planning, Journal of Management Information Systems 3, No. 4 (Spring 1987) 4–38.

[2] N. Arni et al., **(please complete)** Overview of RAD: A Hybrid and Distributed Reasoning Tool, MCC Technical Report No. ACT-RA-098-90 (March 1990) 77 pages.

[3] W.R. Ashby, Principle of the Self-Organizing System, in: H. von Foerster and G. Zopf, Eds., Principles of Self-Organization (Pergamon, Elmsford, NY, 1962).

[4] T. Basar, P.R. Kumar, M. Loui, M. Shaw, Distributed Decision Making, Learning, and Coordination in Organizations with a Hybrid Structure, Research Proposal submitted to the National Foundation of Science (Nov. 1988).

[5] A.B. Baskin, S.C-Y. Lu, R.E. Stepp and M. Klein, Integrated Design as a Cooperative Problem Solving Activity, Proceedings of the Hawaii International Conference on System Sciences (1989) 387–396.

[6] I. Benbasat, B. Konsynski, Introduction to Special Section on GDSS, MIS Quaterly (Dec. 1988) 588–590.

[7] R.W. Blanning, Knowledge, Metaknowledge, and Explanation in Intelligent Organizational Models, Technical Report (Owen Graduate School of Management, Vanderbilt University, Nashville, TN, 1990) 24 pages.

[8] A.H. Bond, The Cooperation of Experts in Engineering Design, in: L. Gasser and M. Huhns, Eds., Distributed Artificial Intelligence, vol. II. (Pitman, London, 1989) 463–486.

[9] T.X. Bui and M. Jarke, Communications Design for Co-oP: A Group Decision Support System, ACM Transactions on Office Information Systems 4 (April 1986) 81–103.

[10] S. Cammarata, D. McArthur and R. Steeb, Strategies of Cooperation in Distributed Problem Solving, Proceedings of the Eighth International Joint Conference on Artificial Intelligence (1983) 767–770.

[11] C. Ching, C. Holsapple, A. Whinston, Reputation, Learning, and Coordination in Distributed Decision-Making Contexts, Organization Science (1990) **vol, pp?**

[12] M.D. Cohen, Artificial Intelligence and the Dynamic Performance of Organizational Designs, in: J. March and R. Weissinger-Baylon, Eds., Ambiguity and Command (Pitman, Marshfield, MA, 1986).

[13] D. Corkill, Hierarchial Planning in a Distributed Environment, Proc. of the Sixth International Joint Conference on Artificial Intelligence (Tokyo, Aug. 1979) 168–175.

[14] W.B. Croft and Lefkowitz, Knowledge-Based Support of Cooperative Activities, Proceedings of the Twenty-First Hawaii International Conference on System Sciences, Vol. III (1988) 312–318.

[15] K.S. Decker, Distributed Problem Solving: A Survey, IEEE Transactions on Systems, Man, and Cybernetyics 17, No. 5 (Sept. 1987).

[16] A. Dennis, J. George, L. Jessup, J. Nunamaker and D. Vogel, Information Technology to Support Electronic Meetings, MIS Quarterly (Dec. 1988).

[17] G. DeSanctis and B. Gallupe, A Foundation for the Study of Group Decision Support Systems, Management Science 33 (1987) 589–606.

[18] E.H. Durfee, V.R. Lesser and D.D. Corkill, Cooperation Through Communication in a Distributed Problem Solving Network, in: M. Huhns, Ed., Distributed Artificial Intelligence (Pitman, London, 1987) 29–58.

[19] C. Ellis, S. Gibbs, and G. Rein, Groupware: Some Issues and Experiences, Communications of the ACM, 34, No. 1, (Jan. 1991) 39–58.

[20] S. Finger, M. Fox, F. Prinz, J. Rinderle, Concurrent Design, Technical Report (Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1990).

[21] M. Fox, An Organizational View of Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics 11 (1981) 70–79.

[22] M. Fox, E. Gardner, S. Safier and M. Shaw, The Role of Architecture in Computer-Assisted Design Systems, Technical Report (Robotics Institute, Carnegie Melton University, Pittsburgh, PA, forthcoming).

[23] M.R. Genersereth, M.L. Ginsberg and J.S. Rosenschein, Cooperation without Communication, Proceedings of the Fifth National Conference on Artificial Intelligence (Philadelphia, PA, 1986) 51–57.

[24] M. Georgeff, Communication and Interaction in Multiagent Planning, Proceedings of the AAAI National Conference (1983).

[25] P. Gray and L. Olfman, User Interface in Group Decision Support Systems, Decision Support Systems, (1989) 119–137.

[26] B. Hayes-Roth, A Blackboard Architecture for Control, Artificial Intelligence Journal 26 (1985) 251–321.

[27] J.H. Holland, Adaptation in Natural and Artificial Systems (University of Michigan Press, Ann Arbor, MI, 1975).

[28] J.H. Holland, Induction: Processes of Inference, Learning and Discovery (MIT Press, Cambridge, MA, 1986).

[29] K.L. Kraemer and J. King, Computer-based Systems for Cooperation Work and Group Decision Making, Computing Survey 20 **(year?)** 115–146.

[30] P. Langley, H. Simon, G.L. Bradshaw, J. Zytkow, Scientific Discovery (MIT Press, Cambridge, MA, 1987).

[31] Laughlin, P.R. and T.A. Shippy, Collective Induction, Journal of Personality and Social Psychology 45, No. 1 (1983) 94–100.

[32] V.R. Lesser and D.D. Corkill, Functionally Accurate, Cooperative Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics 11, No. 1 (1981) 81–96.

[33] H.A. Linstone and M. Turoff Eds., The Delphi Method; Techniques and Applications (Addison-Wesley, Reading, MA, 1975).

[34] P. Lounamaa and J. March, Adaptive Coordination of a Learning Team, Management Science 33 **(year?)** 107–123.

[35] T.W. Malone, Modeling Coordination in Organizations and Markets, Management Science 33 (1987) 1317–1332.

[36] T.W. Malone, What is Coordination Theory ? Proceedings of the AAAI Workshop on Distributed Artificial Intelligence (Lake Arrowhead, CA, 1988).

[37] T.W. Malone and S.A. Smith, Modeling the Performance of Organizational Structures, Operations Research 36 (1988) 421–436.

[38] R.S. Michalski, A Theory and Methodology of Inductive Learning, Artificial Intelligence 20 (1983) 111–161.

[39] M. Minsky, The Society of Mind (Simon and Schuster, New York, 1985).

[40] D. Navinchandra, Case Base Reasoning in CYCLOPS: A Design Problem Solver, in: J. Kolodner, Ed., Proceedings of Case-Based Reasoning Workshop (Morgan Kaufmann Publishers, San Mateo, CA, 1988) 286–301.

[41] H.P. Nii, N. Aiello and J. Rice, Experiments on Cage and Poligon: Measuring the Performance of Parallel Blackboard Systems, in: L. Gasser and M. Huhns, Eds., Distributed Artificial Intelligence, vol. II (Pitman, London, 1989) 319–384.

[42] J.F. Nunamaker, L.M. Applegate and B.R. Konsynski, Computer-Aided Deliberation: Model Management and Group Decision Support, Operation Research 36, No. 6 (1988) 826–847.

[43] J.F. Nunamaker, D. Vogel and B. Konsynski, Interaction

of Task and Technology to Support Large Groups, Decision Support Systems (1989) 139–152.

[44] J.R. Quinlan, Induction of Decision Trees, Machine Learning 1 (1986) 81–106.

[45] H.V.D. Parunak, J. Kindrick and B. Irish, Material Handling: A Conservative Domain for Neural Connectivity and Propagation, Proceedings of the AAAI National Conference (1987) 307–311.

[46] L. Rendell, A New Basis for State-Space Learning Systems and a Successful Implementation, Artificial Intelligence vol. ?, No. 2 (1983) 177–226.

[47] J.S. Rosenschein and M.R. Genersereth, Deals Among Rational Agents, Proceedings of the Ninth International Joint Conference on Artificial Intelligence (1985) 91–99.

[48] A. Sathi and M. Fox, Constraint-Directed Negotiation of Resource Reallocations, in: L. Gasser and M. Huhns, Eds., Distributed Artificial Intelligence, Vol. II (Pitman, London, 1989) 163–194.

[49] R.C. Shank, Dynamic Memory (Cambridge University Press, Cambridge, 1982).

[50] M. Shaw and R. Sikora, A Distributed Problem-Solving Approach to Rule Induction, Technical Report CMU-RI-TR-90-26 (Robotics Institute, Carnegie Mellon University, Pittburgh, PA, Nov. 1990).

[51] M. Shaw and A.B. Whinston, Learning and Adaptation in Distributed Artificial Intelligence Systems, in: L. Gasser and M. Huhns, Eds, Distributed Artificial Intelligence, vol. II (Pitman, London, 1989) 413–430.

[52] M. Shaw, B. Harrow and S. Herman, Distributed Artificial Intelligence for Multi-Agent Problem Solving and Group Learning, Proceedings of the Hawaii International Conference of Systems Science (IEEE Press, 1991).

[53] H.A. Simon, The Sciences of the Artificial (MIT Press, Cambridge, MA, 1982).

[54] H. Simon, G. Lea, Problem Solving and Rule Induction: A Unified View, in: L.W. Gregg, Ed., Knowledge and Cognition (Erlbaum, Potomac, MD, 1974).

[55] R.G. Smith, The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers 29 (1980) 1104–1113.

[56] R.G. Smith and R. Davis, Frameworks for Cooperation in Distributed Problem Solving, IEEE Transctions on Systems, Man, and Cybernetics 11, No. 1 (Jan. 1981) 61–70.

[57] M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning and L. Suchman, Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings, Comm. Ass. Comput. Mach. 30, No. 1 (1987) 32–47.

[58] K. Sycara, Using Case-Based Reasoning for Plan Adaptation and Repair, in: J. Kolodner, Ed., Proceedings of Case-Based Reasoning Workshop (Morgan Kaufman Publishers, San Mako, Ca, 1988) 425–434.

[59] K. Sycara, Multi-Agent Compromise via Negotiation, in: L. Gasser and M. Huhns, Eds., Distributed Artificial Intelligence, Vol. II (Pitman, London, 1989) 119–137.

[60] M. Warglien, Learning by Choosing in a Garbage Can Situation: A Coonectionist Approach, Technical Report (Centre de Sociologie des Organisations, Paris Universita degli Studi di Venezia, Venice, Oct. 1988) 32 pages.

[61] C.C. Woo and F.H. Lochovsky, Supporting Distributed Office Problem Solving in Organizations, ACM Transactions on Office Information Systems (July 1986) 185–204.