Cooperation Through Information Interchange in StormCast

G. Hartvigsen and D. Johansen

Department of Computer Science, University of Tromsø, N-9000 Tromsø, Norway

Abstract. This paper addresses the cooperation between different expert system modules in a networking environment. StormCast – a distributed artificial intelligence application for severe storm forecasting is used as a case to obtain practical results. Two key aspects is investigated, first the representation of knowledge in this kind of environment is outlined. Then the cooperating nature of a group of expert system modules is discussed.

Keywords. Computer cooperation; distributed application; distributed artificial intelligence; expert system; knowledge representation; weather forecasting.

INTRODUCTION

Distributed artificial intelligence (DAI) is concerned with problem solving in which groups solve tasks. Research (e.g., Fikes and Nilsson, 1971; Fahlman, 1974; Sacerdoti, 1974, 1977) has given us an understanding of centralized problem solving. Unfortunately, this is not the case for distributed or group problem solving. Konolige (1982) proposes that both knowledge representation and problem solving techniques in distributed or group problem solving diverse from that required of single agent problem solvers. One particularly important but little understood topic in this area is cooperation, a key issue to be investigated in the StormCast project. The fundament for cooperation is a common understanding of the domain, i.e. the knowledge of the topic which are investigated. The implementation of such aspects calls for appropriate knowledge representation technique(s).

In the StormCast project, a basic assumption is that severe storms can be predicted. The field of weather forecasting offers good prospects for the application of both distributed computing (Johansen, 1989) and artificial intelligence (AI) techniques (Gaffney and Racer, 1983; Racer and Gaffney, 1984; Swetnam and Dombroski, 1985; Elio and De Haan, 1986; Merrem, 1986). Even better prospects are shown by combining these fields into distributed artificial intelligence applications. In the StormCast project we have taken the advantages of this convergence into account.

In this paper, an outline of the current status of the StormCast project is given. Then we discuss the knowledge representation and cooperation in StormCast, a distributed artificial intelligence application for severe storm forecasting.

STORMCAST

StormCast is a distributed artificial intelligence (DAI) application predicting severe storms in the Arctic part of the Northern hemisphere (Hartvigsen and Johansen, 1989). Bond and Gasser (1988) define distributed artificial intelligence as “the subfield of AI concerned with concurrency in AI computations, at many levels”. We consider StormCast to be a DAI application since cooperation between logically separated and concurrent expert systems is necessary to obtain the desirable functionality which is severe storm forecasting. In addition, the monitoring and gathering of raw weather data is distributed in its nature (Johansen, 1989). Forecasting weather in general is inevitably difficult, but the main purpose with the StormCast project is not weather forecasting as a science. Our intention is to study DAI, and we use StormCast as a case to obtain practical results in this area.

In the StormCast project, we have only employed weather data collected from ground installations, and not used results gathered by satellites. Neither have we utilized information received from more traditional equipment which measures the wind speed, temperature, dewpoint, wind intrusion, etc., at different levels beyond the ground. A motivation for this is the requirement of almost real-time response in our application together with high availability, which the satellites, weather balloons and such kind of equipment, do not solve properly yet.

As shown in fig. 1, StormCast is hierarchically organized into two main layers; a data collection layer and a knowledge layer. The data collection layer consists of the monitoring and synthesizing modules (Johansen, 1989). The knowledge layer includes the transmission and expert modules (Hartvigsen and Johansen, 1989). One purpose with this hierarchical architecture is scaleability. Adding or removing new weather domains to the application is straightforward.

The specification of this kind of process structure is complex and domain-dependent, and involves the identification of functional components, their responsibilities and resource requirements, and the relations among them. Relations include communication relations, authority relations which state the significance of the received data, and proximity relations which identify horizontal differentiation among objects. All of these relations are even more complex for the reason of interacting constraints. Together, this information represents the organizational structure of the system (Pattison, Corkill and Lesser, 1987)

.

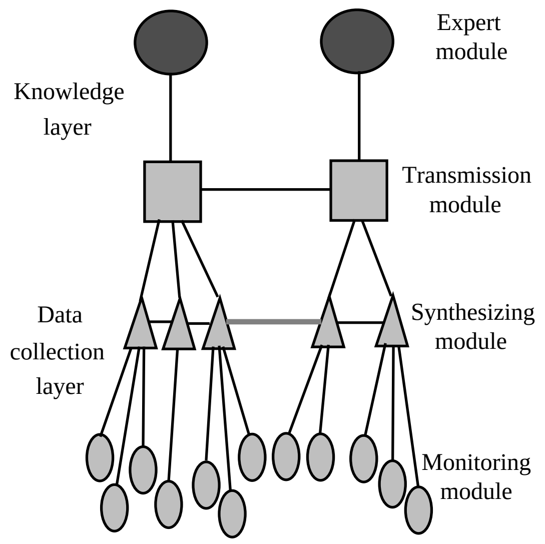

Fig.1.The hierarchical organizational structure of StormCast.

Initially, a prototype weather monitoring application (Johansen, 1989) was designed to be run on the Amoeba distributed system (Mullender and Tanenbaum, 1986) connected through a wide area network (Renesse et al., 1988). This concept was further developed in a previous version of StormCast (Hartvigsen and Johansen, 1989). But since this experimental distributed system is not in widespread use, the application has been re-implemented on another software base where each module is implemented in C or Common Lisp as a UNIX process using current standards as TCP/IP and the X Window system. The hardware consists of Motorola 68030 workstations (Hewlett-Packard 9000/360 CH and Apollo DN3500) using Ethernet-based LAN locally and X.25 for WAN connections. Altogether this makes a proper platform to build distributed applications.

Having modules of a distributed application located on different nodes can be done to improve overall performance, meet functional requirements, utilize present resources, etc., but this might also imply additional problems. Mainly, this is due to the requirement that a distributed application must be transparent, at least in local networking environments. The decentralized nature of a distributed application is to be hidden for the users to achieve this. Problems with global state detection, network partitioning, communication delays in networks, etc., might be sources of problems to meet these transparency requirements. In the StormCast project, our approach to achieve transparency is to simplify whatever wherever possible. Clearly, benefits with a service or extra quality levels of service in a distributed application must compensate for the additional costs with it, and our decisions have been strongly influenced by this.

An important aspect of a distributed application environment is that significant delays in the interprocess communication are commonplace. Having a distributed application in a wide area networking environment makes the delayed effects of the interactions between the modules difficult if not impossible to have a consistent global state of StormCast. Potential problems with global state detection have been solved in StormCast, mainly since we do not need the accurate global state or optimal synchronization to achieve what is expected of this application.

Updates to a replicated database must be synchronized to guarantee consistency, but we do not consider synchronization in StormCast as a fine-granularity problem as in distributed databases or distributed systems. In StormCast we achieve sufficient synchronization and ordering of events by time-stamping the data with real-time values, i.e. temperature monitored at 11.11 p.m. will do as well as 11.12 p.m., a result obtained from a remote expert system one minute ago, and not one second, will do in the local decision process. There is small problems with updates arrived at a node not in time-stamped order since the real time clock is used. Normally, a message from time t_0 might arrive at a node later than a message at time $t_0 + 1$. It is up to the expert system to determine whether the time t_0 is acceptable; if so, the weather prediction is carried out.

There is no problem with information growing over time exhausting available storage at a node. Updates older than a given timestamp no longer needed can be discarded. In addition, old data is also discarded by simply over-writing with the new value and its timestamp. This is a simplification compared to other approaches as (Sarin and Lynch, 1987). Since the distinction between real-time and non real-time approaches often is insignificant, a non real-time approach can be chosen under such circumstances. By a limited (insignificant) reduction in the quality of the reasoning process, a lot of problems might be avoided.

KNOWLEDGE REPRESENTATION

StormCast may be described as a distributed computer control system. This kind of system is extensively using control information. In this case, the system is a meteorological monitoring expert system continuously analysing on-line data sources to predict severe storm. In the design of the expert module, we had to consider the way in which meteorologists generally works. As a consequence, appropriate knowledge representation techniques had to be chosen. This section outlines how knowledge is represented in StormCast, and the local usages of this knowledge is also demonstrated.

Knowledge Representation – a Critical Factor

The fundamental idea of an expert system is the manipulation of knowledge in order to solve problems effectively in a narrow problem area (Cercone and McCalla, 1987). Several questions have to be solved in order to select an appropriate knowledge representation technique, e.g. the structure of knowledge in a knowledge base, encoding of rules in order to infer knowledge, inference control, handling incomplete knowledge, etc. In addition, we have to be aware of the difference between a knowledge representation system and a database. A knowledge representation system in general performs inferencing of some kind in order to answer queries about what is represented, while the database system is limited to retrieving the facts it contains (Duce and Ringeland, 1988).

In StormCast, we utilize several expert modules (i.e. expert systems) which all shall attain such high levels of performance, partly by exchanging knowledge (weather forecasts). To obtain this goal we need to have a good understanding of the knowledge required as well as a suitable tool for the knowledge representation. The severe storm forecasting process is an ongoing process. Normally, each expert module is activated at a fixed time interval. The expert module then issues a request for weather data to its corresponding transmission module which forwards the request to the related synthesizing modules. This request is carried out by a further client-server relationship between the synthesizing modules and their monitoring modules. If necessary, the data is preprocessed by the synthesizing node before it is sent to the corresponding transmission module. Each transmission module maintains a table which is periodically (with period p) updated. The expert module executes periodically every d seconds using the current weather information passed by the transmission module, which in turn receives data from the rest of the application.

It seems that meteorologists often apply declarative knowledge which possibly can be used as a production rule. This has lead us to use a rule based approach in the development of the expert module. Hayes-Roth (1985) argues that rule-based systems constitute the best available means for codifying the problem-solving know-how of human experts. The expert module receives information from its transmission module. Together, the rule-based systems predicting weather in delimited geographical areas and the transmission modules containing and exchanging local and global weather data, constitutes what we refer to as a simple blackboard system (see fig. 2).

As in a traditional blackboard system (Engelmore, Morgan and Nii, 1988), StormCast employs several knowledge sources which are kept separate and independent. But since StormCast is designed to operate in a very wide geographical area, we have chosen to let each expert module has its own transmission module. This reduce the possibility of a complete system breakdown caused by transmission failures. Relevant weather data is multicasted. Therefore, we consider the different transmission modules to be one logical unit – a global database. All communication and interaction among the knowledge sources takes place solely through the blackboard database administrator (i.e. the control function in the transmission modules). Through minimizing the system rendezvous we have limited the complexity of the blackboard framework (Hartvigsen and Johansen, 1989).

As shown in rule148 in the next section, the expert module (i.e. the expert system) employs weather forecasts multicasted by its neighbour expert modules. This means, for example, that the severe storm forecast made in Tromsù, employs results gathered and processed in its neighbour geographical areas (i.e. severe storm forecasts made by the expert modules), as well as data received from its own synthesizing modules. The transmission and classification of data is handled by the transmission module.

The Utilization of Rules in StormCast

In the current version of StormCast, we have built several small and almost identical rule-based expert systems (i.e. expert modules) in order to demonstrate the ability and functionality of distributed artificial intelligence applications. The rule-based expert systems are written in Common Lisp, and are implemented as forward-chaining rule-based programs.

In order to identify each prediction and node we use a name-function to make a easy recognizing and unique name to each prediction. The name consists of the name of the node, current time and date, e.g. no.tromsoe.1112.210789. This makes it easier to identify results received from neighbour nodes and the time the forecasts were made.

In the forecasting process, the expert system (in the expert module) uses, for each variable, one of three different values for the possibility of severe storm; weak, moderate and strong. An initial description is generated from the weather data received from the transmission module. The system is adding this description of the current weather into its working memory by using rules like:

```
(rule015
  if   (> surf-press-in-mb 1009.0)
  then (conclude surf-press strong))

(rule016
  if   (< 1004.0 surf-press-in-mb 1010.0)
  then (conclude surf-press moderate))

(rule017
  if   (< surf-press-in-mb 1005.0)
  then (conclude surf-press weak))

...
```

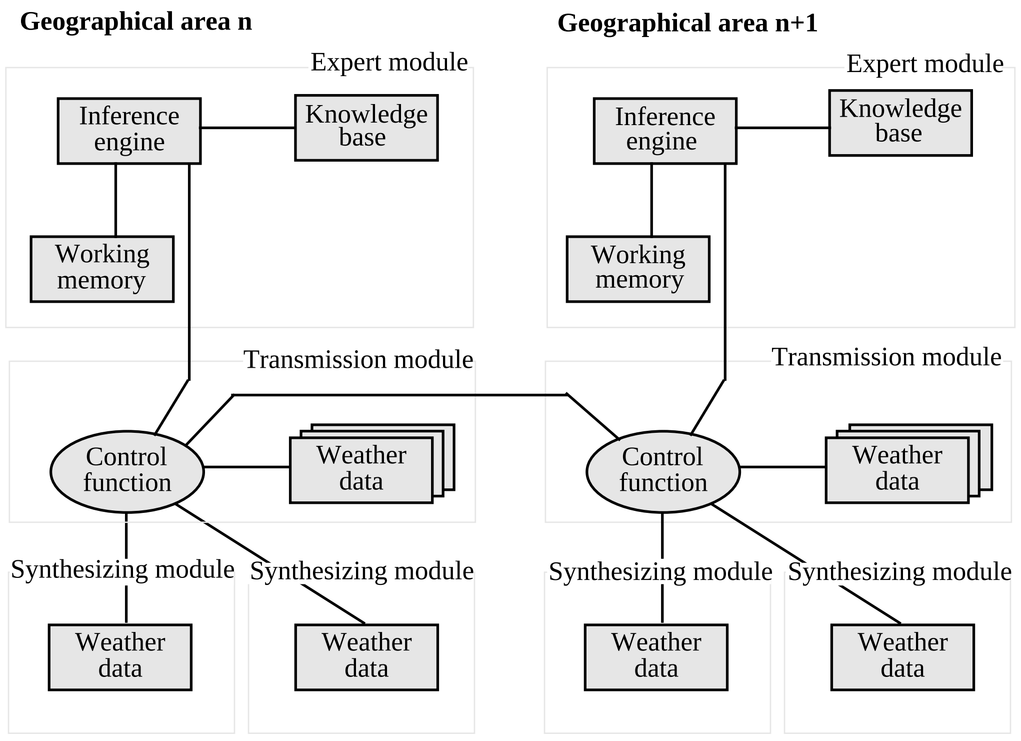

Fig 2. The blackboard framework in StormCast.

```
(rule034
  if   (> surf-dewpoint-in-f 64.0)
  then (conclude surf-press strong))

...

(rule043
  if   (> (- surf-press-in-mb 12h-surf-press-in-mb) 5.0)
  then (conclude surf-press-fall strong))

...
```

In StormCast, parts of the complete application can be unavailable due to for instance network failures, the expert system use control rules to ensure that the forecast can be done. In the current version of the system, the transmission module uses the number -9999 to indicate lack of the original data. As the rules which generate the initial description uses real numbers, the description value may be wrong. Therefore, the expert system uses control rules in order to ensure that the initial description is not to optimistic, e.g.:

```
(rule082
  if   (= surf-press-in-mb -9999)
  then (conclude surf-press pessimistic-value))
```

The pessimistic-value is set to the most pessimistic prediction among the existing data, and is generated by rules like:

```
(rule085
  if    (or (and (> surf-press-in-mb -9999) (equal surf-press
strong))
          (and (> surf-dewpoint-in-f -9999)
              (equal surf-dewpoint strong))
          (and (> 12h-surf-press-in-mb -9999)
              (equal surf-press-fall strong))
 ...

then (conclude pessimistic-value strong))
```

A better solution than the employment of the number -9999, is to use the time-tag in the name, and append to each value, a certainty factor that depends on the age of the information. Furthermore, the linguistic variables (weak, moderate and strong) may be substituted by more exact numerical variable values. This kind of approach will be examined at a later stage in the project.

The current weather data is analysed in order to reach a final conclusion through rules like:

```
(rule142
  if (and (equal surf-press strong)
         (equal surf-press-fall strong)
         (equal surf-dewpoint strong))
  then (conclude severe-storm strong))

...

(rule148
  if   (and (equal surf-press-fall strong)
           (or (equal surf-press strong) (equal surf-press moder-
ate))
           (or (equal surf-dewpoint strong)
              (equal surf-dewpoint moderate))
           (or (equal neigh01-severe-storm-threat strong)
              (equal neigh02-severe-storm-threat strong))
  then (conclude severe-storm strong)

...
```

The current version of the expert system is neither capable nor intended to make a forecast acceptable for public use. However, it shows the ability of using this kind of system architecture in weather forecasting, i.e. it is meant more to illustrate a structure that could serve as the basis for a DAI application than to be operational.

In StormCast, we define information to be preprocessed data inferred from the expert systems. The expert systems (i.e. the expert modules) cooperate through exchanging results from the forecasting processes. This will be further discussed in the next chapter. The weather forecast information is multicasted by the transmission module to the immediate neighbours (Hartvigsen and Johansen, 1989).

In the current version, the expert/transmission module only exchanges the final conclusion, as shown in rule148 in this section. An extension would be to include part-solutions in the multicasted information.

Parallel Execution of Rule-Based Expert Systems

The different expert modules execute in real parallel. Through the use of the Common Lisp and the C languages as well as standard UNIX and TCP/IP functions, the weather forecasting simultaneously occur in geographical separated locations.

In addition, other solutions to achieve parallel execution exists. An alternative would be to utilize parallel programming tools like Multiplisp (Halstead, 1985). However, Forgy et al. (1977) have suggested that rule-based systems have fundamental limits on their parallelism potential.

General Suggestions for Knowledge Representation

The results gained from the implementation of the knowledge layer (the expert module and the transmission module) in StormCast have so far indicated some general proposals concerning parallel execution and the choice of representation technique.

The decision process in order to chose an appropriate knowledge representation tool shall include the consideration of:

- o Data structures: One has to evaluate the flexibility, which means the independency of different processes running in parallel. In StormCast, we required maximum flexibility, i.e. the different processes have to be able to proceed without receiving the information from the others. In addition, the system must easily be extended or re-implemented. These requirements are possible through the combination of the Common Lisp and the C languages.

- o Synchronization primitives: The utilization of 3'rd generation languages and current operation system services requires that most coordination mechanisms have to be implemented from scratch. The transmission module in StormCast is implemented in C and guarantee consistency by using semaphores (Dijkstra, 1968).

The execution speed of StormCast might be increased by dividing the expert system (in the expert module) into different blocks (through the utilization of a top-down approach). By running these blocks on locally separated processors, a higher degree of speed-up might be gained.

The diversification into several expert modules capable of, if necessary, make the weather forecast on their own have made it possible to take advantages from techniques from single agent problem solvers. The group solving aspects are handled by a dedicated function, i.e. the transmission module. The results so far received, indicate a connection between the degree of local autonomy and the complicity of the data structures and synchronization primitives.

COOPERATION

In the previous section, we showed how knowledge is represented in StormCast and how rules can be applied locally on each expert module. In this section, we investigate cooperation issues among the different expert modules (agents) in StormCast. Agents in a DAI concept have the ability to improve both coordination and coherence by controlling what, how, and when they communicate with one another (Bond and Gasser, 1988). In StormCast, the communication improves an agent's local decision making by providing information generated by other agents. This implies a global agreement of the aspects of the cooperation. In a DAI concept, however, communication between agents is just another problem-solving tool that may be used poorly or effectively in order to provide the basis for effective cooperative problem solving (Cammarata, McArthur and Steeb, 1983).

In the StormCast project, we leave the field of networking in the traditional sense and concentrate on the communication nature at the highest layer. This means that we have focused on what and when information interchange occur or is necessary, rather than how the transmission take place on the lowest level(s). Acceptable solutions exist even commercially today and we find it more interesting to investigate difficulties facing a cooperating nature, distributed problem solving groups, and the necessary cooperative strategy in this environment.

Cooperation in DAI

Cooperation in DAI is based on communication in one way or another. Cooperation is a fundamental aspect in DAI, and a lot of research has been done (Erman and Lesser, 1975; Smith and Davis, 1981; Cammarata, McArthur and Steeb, 1983; Durfee and Lesser, 1987; Durfee, Lesser and Corkill, 1987a 1987b; Durfee, 1988). Cooperation in DAI means to have expert systems working together towards solving (a) common problem(s). This calls for the development of cooperative interaction mechanisms that allow multiple expert systems to participate in a teamwork.

The introduction of computers as agents in human problem solving has raised new problems that stress issues in cooperation. In weather forecasting, for example, the information interchange is made easier because the meteorologists have an understanding of each other. Human-computer interactions are often disturbed by misunderstandings caused by an inadequate view of agent behaviour (Oberquelle, Kupka and Maass, 1983). We find analogue problems related with computer-computer communication because computers have primitive, if any, abilities to take advantages of other computers. The employment of artificial techniques, and thereby explicitly represent the understanding, has made it possible to extend this kind of feature.

The simplifying approach

In StormCast, we have designed what we refer to as the simplifying approach. First, we have avoided problems with global state detection, synchronization, etc., by timestamping data with real-time values. Second, our approach is based on the philosophy of exchanging partial solutions. A solution is reached by an iterative process of exchanging preliminary, partial solutions. (The solutions, i.e. severe storm forecasts, in StormCast are partial for the receivers but not for the senders.) Unlike problems which have to be managed in the arena of distributed problem solving (DPS), for example in the FA/C approach by Lesser and Corkill (1981), the multiagent (MA) systems, which is the research arena the simplifying approach has to control, does not need to converge on one overall network solution. Even though this reduce the knowledge needed at each node (module) in the case of predicting each others next steps, both arenas seem to explore complex real-time cooperating approaches. It seems that the utilization of such approaches are not necessary in all situations, as in weather forecasting.

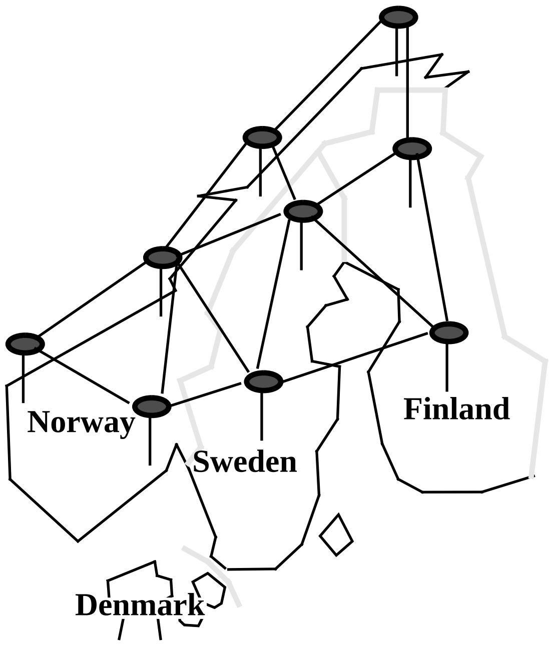

Fig 3. The geographical location of the expert modules in StormCast in the Northern part of Norway.

The simplifying approach represents a suitable solution for weather forecasting. Since the data, as indicated in fig. 3, is exchanged over a very wide geographical area, it is not recommendable to have a tight connection between the expert modules. However, each expert module must contain information concerning the others (organizational knowledge), e.g. geographical location in order to give the received results the correct weight in its own forecasting process. In this way, an expert module becoming isolated does not affect the rest of the system as it would if the forecasting process in other parts of the system were delayed or even halted as a result of this accident.

StormCast may be described as a set of cooperating modules (agents) which continuously are collecting and processing weather data from a fixed geographical area. In each geographical area, there is an expert module (i.e. expert system) responsible for the prediction of, in this case, severe storms (see fig. 3). Each expert module has the knowledge and intelligence needed to make a severe storm forecasting. This severe storm forecast is based on the results achieved from the monitoring modules in their own area. In this way, the problem solving emphasizes intelligent local control of each expert module (problem solver). A network of these expert modules performs distributed problem solving by cooperating as a team to solve the same problem more properly. The cooperation requires that each expert modules knows which solutions to communicate. Such networks are typically utilized in this kind of distributed sensor networks (Lesser and Erman, 1980; Smith, 1980).

The different expert modules are primarily concerned with the forecast of severe storm in its own region. Each problem solver is thus self-interested because the local data is considered to be most important in the problem solving. Cooperation among self-interested problem solvers is based on the assumption that this is in their own interest (Durfee, Lesser and Corkill, 1987a). In weather forecasting the meteorologists working in regional institutes made their predictions upon their local weather observations and data and weather forecasts received from other institutes. The employment of the results received from other institutes depends on how the results fit their own results. The task of obtaining such coherent cooperation is a considered to be a difficult task (Lesser and Corkill, 1981; Davis and Smith, 1983).

CONCLUSIONS

Research in cooperation and coordination among independent nodes involves both distributed artificial intelligence, distributed operating systems and studies of cooperation in natural systems. In this paper, we have mainly focused on concepts within DAI. StormCast is a distributed artificial intelligence application for severe storm forecasting (Hartvigsen and Johansen, 1989) which has functioned as a means of gaining real experience with DAI.

In the StormCast project, our main strategy in the design and implementation of the system have been simplification – to keep the design and the source code as simple as possible without main losses in the functionality. The major motivation for this has been the transparency requirements. A modular design together with the utilization of de facto industrial standards has appeared to be sufficient in the task of weather forecasting in a distributed artificial intelligence concept. In addition, through the use of our simplifying approach, we have avoided problems with global state detections, synchronization, etc., which in term influences the transparent view the users have of the application.

One of the main problems to face in distributed artificial intelligence applications is the coordination of cooperating nodes. A lot of effort has been spent on the coordination problem, and several approaches to improving coordination among cooperating nodes have been presented. A common denominator for the approach presented seems to be real-time computing, which in our experience is a very resource intensive solution. Therefore, we have used a non real-time approach in order to reduce the complexity, but maintain the functionality.

However, even if our choice of system architecture, a data-driven forward-chaining expert system module together with the simple blackboard system (the knowledge layer), has shown some of the possibilities with this kind of application, still much work remain to be done to obtain a commercial application. Further research might include the study of a more complex rule-based expert system that includes both forward and backward chaining, certainty factors, possibly frame-based reasoning, etc., and the diversification of the blackboard in small specialized parts, each guarded by its own monitor. This may in turn increase both the qualitative and the quantitative system performance.

ACKNOWLEDGEMENT

Availability of the distributed system Amoeba has stimulated the development of distributed applications. Discussions with other members of the TRACE project at the University of Tromsù have been valuable, especially comments from O.J. Anshus.

REFERENCES

Bond, A., and L. Gasser (1988). An analysis of problems and research in DAI. In A. Bond and L. Gasser (Eds.), Readings in Distributed Artificial Intelligence. Morgan Kaufmann, Los Altos, California. pp. 3-36.

Cammarata, S., D. McArthur, and R. Steeb (1983). Strategies of cooperation in distributed problem solving. In Proceedings of the 8th International Joint Conference on Artificial Intelligence. Morgan Kaufmann, Los Altos, California. pp. 767-770.

Cercone, N., and G. McCalla (1987). What is knowledge representation? In N. Cerone and G. McCalla (Eds.), The Knowledge Frontier. Essays in the Representation of Knowledge. Springer-Verlag, New York. pp. 1-43.

Davis, R., and R.G. Smith (1983). Negotiation as a metaphor for distributed problem solving. Artificial Intelligence, 20, 63-109. Dijkstra, E.W. (1968). The structure of "THE" multiprogramming system. Communications of the ACM, 11 (5), 683-696.

Duce, D., and G. Ringeland (1988). Background and introduction. In G.A. Ringeland and D.A. Duce (Eds.), Approaches to Knowledge Representation: An Introduction. Research Studies Press, Letchworth, Hertfordshire, England. pp. 1-11.

Durfee, E.H. (1988). Coordination of Distributed Problem Solvers. Kluwer, Boston.

Durfee, E.H., and V.R. Lesser (1987). Using partial global plans to coordinate distributed problem solvers. In Proceedings of the 10th International Joint Conference on Artificial Intelligence. Morgan Kaufmann, Los Altos, California. pp. 875-883.

Durfee, E.H., V.R. Lesser, and D.D. Corkill (1987a). Cooperation through communication in a distributed problem solving network. In P.M. Huhns (Ed.), Distributed Artificial Intelligence. Pitman, London. pp. 29-58.

Durfee, E.H., V.R. Lesser, and D.D. Corkill (1987b). Coherent cooperation among communicating problem solvers. IEEE Transactions on Computers, C-36, 1275-1291.

Elio, R., and J. De Haan (1986). Representing quantitative and qualitative knowledge in a knowledge-based storm-forecasting system. International Journal of Man-Machine Studies, 25, 523-547.

Engelmore, R.S., A.J. Morgan, and H.P. Nii (1988). Introduction. In R.S. Engelmore and A.J. Morgan (Eds.). Blackboard Systems. Addison-Wesley,Wokingham, England. pp. 1-22.

Erman, E.D., and V.R. Lesser (1975). A multilevel organization for problem solving using many, diverse, cooperating sources of knowledge. In Proceedings of the 4th International Joint Conference on Artificial Intelligence. Morgan Kaufmann, Los Altos, California. pp. 483-490.

Fahlman, S. (1974). A planning system for robot construction tasks. Artificial Intelligence, 5, 1-49.

Fikes, R.E., and Nilsson, N.J. (1971). Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2, 189-208.

Forgy, C., A. Gupta, A. Newell, and R. Wedig (1977). Parallelism in artificial intelligence problem solving: A case study of Hearsay II. IEEE Transactions on Computers, C-26 (2), 98-111.

Gaffney, J.E., and I.R. Racer (1983). A learning interpretive decision algorithm for severe storm forecasting support. Automating Intelligent Behavior: Applications and Frontiers. Proceedings of the 1983 Trends and Applications Conference at National Bureau of Standards (25-26 May 1983). IEEE Comput. Soc. Press. pp. 278-284.

Halstead, R. (1985). Multilisp: A language for concurrent symbolic computation. ACM Transactions on Programming Languages and Systems, 7 (4), 501-538.

Hartvigsen, G., and D. Johansen (1989). StormCast – A distributed artificial intelligence application for severe storm forecasting. In M.G. Rodd and T.L. d'Epinay

(Eds.), Distributed Computer Control Systems 1988. Proceedings of the Eight IFAC Workshop (Vitznau, Switzerland, 13-15 September 1988). Pergamon Press, Oxford, England. pp. 99-102.

Hayes-Roth, F. (1985). Rule-based systems. Communications of the ACM, 28 (9), 921-932.

Johansen, D. (1989). Weather forecasting – distributed in nature. Network Information Processing Systems. Proceedings of the IFIP TC6/TC8 Open Symposium (Sofia, Bulgaria, 9-13 May 1988). North-Holland, Amsterdam. pp. 197-203.

Konolige, K. (1982). A first-order formalization of knowledge and action for a multi-agent planning system. In J.E. Hayes, D. Michie, and Y.-H. Pao (Eds.). Machine Intelligence 10. Wiley, New York. pp. 41-73.

Lesser, V.R., and D.D. Corkill (1981). Functionally-accurate, cooperative distributed systems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11 (1), 81-96.

Lesser, V.R., and L.D. Erman (1980). Distributed interpretation: a model and experiment. IEEE Transactions on Computers, 29 (12), 1144-1163.

Merrem, F.H. (1986). Two expert systems used in weather forecasting. Proceedings of Expert Systems in Government Symposium (McLean, VA, USA, 22-24 Oct. 1986). IEEE Comput. Soc. Press, Washington, DC, USA. pp. 458-459.

Mullender, S.J., and A.S. Tanenbaum (1986). The design of a capability-based distributed operating system. The Computer Journal, 29 (3), 289-299.

Oberquelle, H., I. Kupka, and S. Maass (1983). A view of human-machine communication and co-operation. International Journal of Man-Machine Studies, 19, 309-333.

Pattison, H.E., D.D. Corkill, and V.R. Lesser (1987). Instantiating descriptions of organizational Structures. In M.N. Huhns (Ed.), Distributed Artificial Intelligence. Pitman, London. pp. 59-96.

Racer, I.R., and J.E. Gaffney (1984). The potential role of artificial intelligence/expert systems in the warning and forecast operations of the National Weather Service. Proceedings of Expert Systems in Government Symposium (McLean, VA, USA, 22-24 Oct. 1986). IEEE Comput. Soc. Press, Washington, DC, USA. pp. 578-586.

Renesse, R. van, H. van Staveren, J. Hall, M. Turnbull, B. Jansen, J. Jansen, S. Mullender, D. Holden, A. Bastable, T. Fallmyr, D. Johansen, Sj. Mullender, and W. Zimmer (1988). MANDIS/Amoeba: A widely dispersed object-oriented operating system. In R. Speth (Ed.), Research into Networks and Distributed Applications. Elsevier, Amsterdam. pp. 823-831.

Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 5, 115-135.

Sacerdoti, E. (1977). A structure for plans and behavior. Elsevier, North-Holland, New York.

Sarin, S.K., and N.A. Lynch (1987). Discarding obsolete information in a replicated database system. IEEE Transactions on Software Engineering, SE-13 (1), 39-47.

Smith, R.G. (1980). The contract-net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, C-29 (12), 1104-1113.

Smith, R.G., and R. Davis (1981). Frameworks for cooperation in distributed problem solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11 (1), 61-70.

Swetnam, G.F., and E.J. Dombroski (1985). An expert system to support the forecasting of upslope snow storms. Proceedings of Expert Systems in Government Symposium. IEEE Comput. Soc. Press, Washington, DC, USA. pp. 567-572.