A FRAMEWORK FOR DYNAMIC KNOWLEDGE EXCHANGE AMONG INTELLIGENT AGENTS

Authors

**Yilmaz Cengeloglu, University of Central Florida,
Department of Electrical and Computer Engineering,
Orlando, FL, 32816, E-mail: *yil@engr.ucf.edu***

**Soheil Khajenoori, Assoc. Prof., Embry-Riddle Aeronautical University,
Department of Computer Science,
Daytona Beach, FL, 32114, E-mail: *soheil@erau.db.erau.edu***

**Darrell Linton, Assoc. Prof., University of Central Florida,
Department of Electrical and Computer Engineering,
Orlando, FL, 32816, E-mail: *dgl@engr.ucf.edu***

ABSTRACT

An intelligent agent is an object with its own knowledge and information base. Each intelligent agent acts in parallel with other intelligent agents and cooperates with a selected set of other agents to achieve a common set of goals. In a dynamic environment, intelligent agents must be responsive to unanticipated conditions. When such conditions occur, an agent may have to stop a previously planned and scheduled course of actions and replan, reschedule, start new activities and initiate a new problem solving process to successfully respond to the new conditions. Problems occur when an intelligent agent does not have enough knowledge to properly respond to the new situation. A framework for dynamic knowledge exchange among intelligent agents has been proposed to allow an intelligent agent to react to knowledge deficiency. Hence, using the proposed framework new knowledge can be transferred when an intelligent agent is unable to solve the problem using its own knowledge. Once the knowledge has been transferred, the intelligent agent can either keep the transferred knowledge permanently or remove it after the transferred knowledge has been used to solve the problem.

1. INTRODUCTION

1.1. Distributed Problem Solving and Objective

A Distributed Problem Solving (DPS) system can be characterized as a group of individual agents running and cooperating with other agents to solve a problem. As dynamic domains such as air traffic controlling and automated factories are continuing to grow in complexity, it becomes more difficult to control the behavior of agents in these domains where unexpected events can occur. In recent years, there has been considerable growth of interest in the design of intelligent agent architectures for dynamic and unpredictable domains. Most of today's intelligent agent architectures are limited to performing pre-programmed or human assisted tasks. In order to be more useful in complex real world domains, agents need to be more robust and flexible. Because of the limited knowledge resources agents may have inadequate problem solving capabilities. They need to learn how to respond promptly to unexpected events while simultaneously carrying out their pre-programmed task.

It is highly possible that an agent with a responsibility in the dynamic environment faces unexpected events. In order to be responsive, the agents should have enough knowledge to deal with the unexpected events. If an agent is not able to deal with a particular event on its own, it can take the following actions :

1) Learn how to solve the problem by experimenting with different solution strategies.
2) Let some other knowledgeable agent solve the problem and then use the results.
3) Learn how to solve the particular problem by acquiring the necessary knowledge from other agents capable of solving the problem.
4) Ignore the unexpected event.

The objective of this research is to investigate and recommend a framework to support distributed problem solving for actions 2 and 3 listed above.

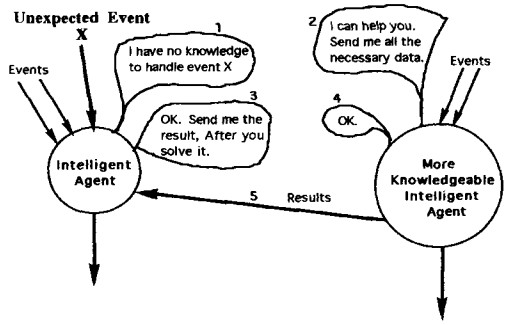

Figure 1. Allowing Another Agent To Solve The Problem.

Figure 1 illustrates problem solving by another agent (i.e., action 2). There are circumstances when this approach may not be efficient; examples may include, excessive data transfer required to solve the problem or the agent may face the same problem over and over again. Under these circumstances, a better solution could be to learn from other agent(s). Figure 2 illustrates how this process might work.

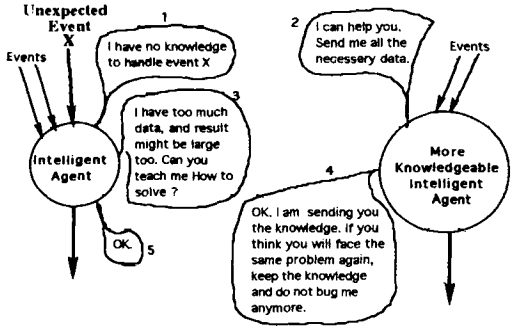

Figure 2. Learning and Teaching Between Intelligent Agents

According to the proposed framework, each agent in the system can learn (permanently or temporarily) how to deal or handle an event for which previous knowledge was not available. This is accomplished by allowing each agent to transfer its knowledge to the other agents rather than solving the problem on their behalf. These abilities play a very important role in the real time distributed problem solving systems because of the dynamic behavior of the environment.

In this research, we address two important issues :

1) How individual agents should be interconnected so that their capacities are efficiently used and their goals are accomplished effectively and efficiently.

2) How the knowledge transfer should take place among agents to allow them to respond successfully to unexpected situations.

1.2. An Overview of DPS Systems

A number of Distributed Problem Solving systems are reported in the literature. Here we provide an overview of some of these systems.

RT-1 architecture [1] is a small-scale, coarse-grained, distributed architecture based on the blackboard model. It consists of a set of reasoning modules which share a common blackboard data and communicate with each other by "signaling events". The reasoning modules operate asynchronously and run in parallel. Blackboard data, shared by all reasoning modules and blackboard, is physically distributed with each module in the distributed multiprocessing environment. The class definitions of various domain dependent object reside on the blackboard. The blackboard also holds information about goals, plans, status, current problem-solving strategy. This system is implemented in LISP.

PRAIS (Parallel Real-Time Artificial Intelligence Systems) [2] is an architecture for real-time artificial intelligence system. It provides coarse-grained parallel execution based upon a virtual global memory. PRAIS has operating system extensions for fact handling and message passing among multiple copies of CLIPS.

MARBLE (Multiple Accessed Rete blackboard linked Experts) [3] is a system that provides parallel environment for cooperating expert systems. Blackboard contains facts related to the problem being solved and it is used for communication among expert systems. Each expert shell in the system keeps a copy of the blackboard in its own fact base. Marble has been used to implement a multi-person blackjack simulation.

AI Bus [4] is a software architecture and toolkit that supports the construction of large-scale cooperating systems. An agent is the fundamental entity in the AI bus and communicates with other agents via message passing. An agent has goals, plans, abilities and needs that other agents used for cooperation. An agent can keep a list of most recently used capabilities and interests of other agents. Using the list, an agent can determine which agent can be most useful to solve the current problem that the agent is facing. Agents share a blackboard which is used to keep information resulting from matching events and current and past conditions.

The work by Raulefs, P. [5] attempts to solve problem that is replaning and rescheduling when unanticipated conditions occur while an agent is running, in blackboard architecture. Knowledge sources are specified with situations which consist of assertion about blackboard state and a description of the time period over which the assertion holds. Situations are trigger conditions, preconditions, while conditions, while effect, termination condition, abnormal termination effect and post effect. This mechanism applied to the real-time control of manufacturing cell using blackboard architecture.

GBB (Generic Blackboard) [6,7] is toolkit for developers needs to construct a high-performance blackboard based applications. The focus in GBB is increasing the efficiency of blackboard access, especially for pattern-based retrieval. GBB consist of different subsytems : a blackboard database development subsystem, control shells, knowledge source representation languages and graphic displays for monitoring and examining blackboard and control components. GBB is an extension of Common Lisp and CLOS (Common Lisp Object System). It allows knowledge sources to be written in any language callable by the developer's Common Lisp Implementations. The blackboard consists of hierarchical structure called spaces and blackboard objects called units. The application programmer can write its own control shell and integrate to

the GBB or can choose a generic control supplied by GBB. GBB has being developed for DVMT (Distributed vehicle Monitoring Testbed) and currently is commercial product.

GEST (Generic Expert System Tool) [8] has been developed by Georgia Tech Research Institute. The main components of the GEST are the central blackboard data structure, independent experts or knowledge sources and the control module. The blackboard data structure holds the current state of the problem solving process. It is also common communication pathway among knowledge sources. Blackboard data structure holds frames, production rules, facts, and procedures. The control module is a single knowledge source, implemented as a GEST expert system. Control has several tasks: 1) It determines which knowledge sources can contribute to the problem solving; 2) It is responsible for checking trigger conditions; 3) It determines knowledge sources that are most desirable to activate. Every action in the GEST is time-stamped. This provides temporal reasoning and backtracking reference point for hypothesis generation and testing paradigms. Time-stamping also allows what-if capability. GEST incorporates a number of knowledge representation techniques and offers several methods for handling uncertainty, a truth maintenance facility, and tools for checking knowledge base consistency. GEST has been implemented in LISP.

The work by Taylor, J.[9] is a framework for running cooperating agents in a distributed environment. This framework supports the Intelligent Computer Aided Design System (ICADS) projects. The core of ICADS is connected to the blackboard control system. Blackboard control system has two main components; message handler and conflict resolver. They are implemented as expert systems using CLIPS shell. Multiple hierarchies of connections among agents is supported by this framework.

MACE (Multi-Agent Computing Environment) [10] has been used to made lower-level parallelism and to build higher-level distributed problem solving architecture. The main components of MACE are agents. Agents in MACE run in parallel, and communicate through messages. Agents provide optional representation and reasoning capabilities. There is no blackboard in the system, agents communicate only via messages. Message can be sent to individual agent, to group of agents, or all agents. Agent Description Language (ADL) is used for describing agents.

CAGE and POLIGON [11] have been developed at Stanford University. They are two different frameworks for concurrent problem solving. CAGE is a conventional blackboard system which supports parallelism at knowledge sources level. The knowledge source, the rules in the knowledge source, or clauses in a rule can be executed in parallel. CAGE is a shared memory multiprocessing system. POLIGONs' functionality is similar to CAGE. POLIGON has been implemented using distributed memory multiprocessing architecture whereas CAGE was implemented using shared memory multiprocessing architecture.

Hearsay-II [12,13,14] is a speech understanding system developed at Carnegie-Mellon University. Hearsay-II provides a framework that different knowledge sources cooperate to solve a problem. Each knowledge source has two major components: 1) preconditions that are used for finding subset hypotheses and 2) action which is a program for applying the knowledge to stimulus frame and making changes to the blackboard. The blackboard is utilized as: 1) storage area for results generated by search process, 2) communication channel among knowledge sources, 3) a database for the system scheduler, and 4) storage area for debugging results.

CASSANDRA architecture [15] is based on multi-level blackboard architecture. The principal component of the architecture is the level manager which is an independent, autonomous unit in a system. Each level manager has a set of knowledge sources, local controller, local database and communication port. There is a communication channel between each level manager which determines the relationships among level managers. All communication channels among level managers must be pre-defined. Dynamic creation of communication channel is not permitted, this prevents the level manager to set a communication link to other level manager(s) at run time. The basic communication model is asynchronous which is sending data at irregular intervals, rather than at clocked times. A communication channel can be either input or output, bi-directional channel is not supported. Each level manager can be connected to many level managers by setting up a communication channel to other level manager(s). Level managers can send and receive two types of messages: Problem Solving Message which is solution of the current problem and Control Solving Message which is relevant to the control of the problem.

Control of traffic signals by a distributed processor is proposed by N. V. Findler [16] for improving traffic flow. In this research, each processor controls an intersection using an expert system. Each expert system contains a different knowledge base associated with the specific intersection. Each processor communicates directly with the four adjacent processors. The information transmitted among processors could be raw data, processed information or expert idea. Prototype system has been implemented in LISP. A traffic simulator and traffic scenario generator has been developed to test the system.

Significant work has been directed to develop Knowledge Interchange Formats (KIF) and Knowledge Query and Manipulation Languages (KQML) [17,18] by Stanford University. KIF is a computer-oriented language for the interchange of knowledge among disparate programs that is written by different programmers, at different times, in different languages. KIF is not a language for the internal representation of knowledge. When a program reads a

knowledge base in KIF, it converts the knowledge into its own internal form. When the program needs to communicate with another program, it maps its internal data structures into KIF. KQML messages are similar to KIF expressions. Each Messages in KQML is one piece of a dialogue between the sender and receiver programs.

1.3. Problem Statement and Discussion

Artificial Intelligence approaches to real time applications are becoming more attractive. An intelligent agent operating in real time environment will find its reasoning and other actions constrained by limitation of time, information, and other critical resources. These agents have to interact with dynamically changing and partially unknown environments. Nonetheless, these agents must be able to give respond to unanticipated changes and events occurring in their operational environment. When unexpected situations occurs, agents must know what actions need to be taken. If the agents are not able to deal with the unexpected situation, they should seek help from other agents in the system to resolve the problem.

The review of current literature in distributed problem solving reveals that most related systems or architectures proposed are limited only to the exchange of problem related facts among the knowledge sources (i.e., agents).

Knowledge exchange among intelligent agents should be done in a manner such that: 1) it does not effect the execution of either receiver or sender, and 2) an intelligent agent should be able to keep the knowledge temporarily or permanently. In other words, knowledge exchange among agents should allow for a form of learning to be accomplished.

In this research, we propose a framework based on distributed blackboard model of problem solving which allows for dynamic knowledge exchange among intelligent agents. The distinguishing characteristic between this framework and existing architectures is capability of dynamic knowledge exchange among intelligent agents. Under this framework, an intelligent agent is capable of transferring knowledge to the other intelligent agents or seeking help from other agents to solve a particular problem that the agent is unable to solve using its own knowledge. We have implemented a prototype system using multiprocessing techniques and operating system facilities on a computer system that has a single processor. Based on the proposed framework, groups of agents run in parallel and cooperate with each other to solve a problem. Each agent can use different problem solving strategies. Most importantly, agents can exchange knowledge (e.g, rules , facts, commands, etc.) dynamically without interrupting the tasks that they are performing.

2. DESCRIPTION AND DISCUSSION OF THE PROPOSED FRAMEWORK

2.1. Overall Architecture of The Framework

The overall architecture of the proposed framework is based on the blackboard model of distributed problem solving. Figure 3 shows the overall architecture of the framework. The framework has four main components:

- Intelligent agents,
- Control,
- Main Blackboard.

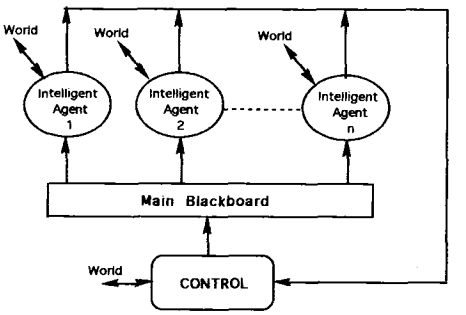

Figure 3. A Communication Network Among Intelligent Agents

An intelligent agent is any process that runs in parallel with other process(es) and has the ability to use the communication protocol to communicate and cooperate with a selected set of other agents to achieve a common set of goals. Intelligent agents can be distributed on different computers, or be on a single computer using multiprocessing techniques or a combination of both. In addition, processing ability of an intelligent agent can vary from very simple functions to very complex behavior. Intelligent agents have the ability to set up a communication link to other intelligent agents or applications. All agents use the blackboard for any form of data transfer among them. The framework does not require the agents to have one-to-one communication links except under special circumstances, such as continuous data transfer between two agents. Intelligent agents are able to transfer facts, rules and commands dynamically.

Control is the core element of the framework. It organizes the blackboard and broadcasts its content when necessary. The main goal of control is to synchronize the framework and handle the blackboard related requests. Framework is synchronized only when changes occur on the blackboard. When there is a different priority among the intelligent agents, control handles the intelligent agents' requests depending on the agent's priority.

The blackboard is a data structure that can be shared by all intelligent agents simultaneously. In the following

sections each component of the proposed framework is explained in detail.

2.2 Intelligent Agent

An intelligent agent is an object with its own knowledge and information base. An intelligent agent's information base is a structure holding facts and knowledge about the part of the system for which that intelligent agent is responsible. An information base is also known as a blackboard.

An intelligent agent has a number of elements which allow it to communicate with other agents to achieve its goals. Figure 4 shows the internal elements of an intelligent agent and their interrelationship. The *Knowledge Source* provides intelligence to the agents, *Reader* receives data from outside the agent and distributes the data to the other elements of the agent, *Sender* transmits the internal data to the outside world, and *Local Blackboard* holds a copy of the main blackboard which contains facts and knowledge.

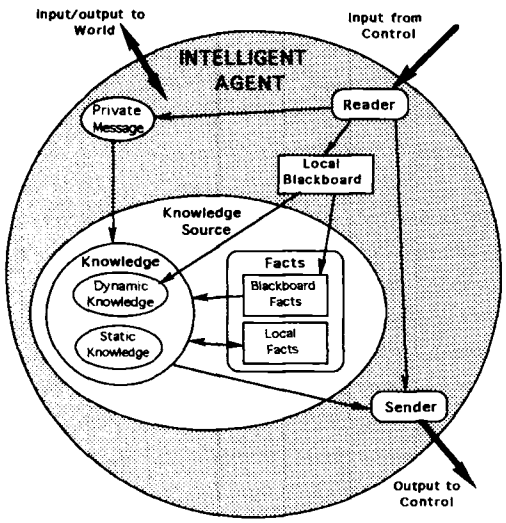

Figure 4. Intelligent Agent

The knowledge source consists of a static knowledge base, dynamic knowledge base, blackboard facts and local facts. A static knowledge base contains rules that are not changed during the program execution. Conversely, the content of a dynamic knowledge base can be changed with the execution of knowledge exchange process. A dynamic knowledge base allows intelligent agents to exchange knowledge when necessary; rules can be deleted and/or inserted into the dynamic knowledge base at run time. This process is termed dynamic knowledge exchange. Facts and rules that need to be broadcasted must be placed by the sender agent on the main system blackboard via the control mechanism. This means under the proposed framework, intelligent agents do not have one-to-one communication links. Any message that needs to be broadcasted must be placed on the main system blackboard.

An intelligent agent also has input/output communication links to the world outside of the application framework. This provides a mechanism to easily integrate the agents with other systems or applications.

2.3. Control

The control process is the most important part of the framework. The control organizes the blackboard and broadcasts its content to the intelligent agents. Figure 5 shows internal component of the control process and their inter relationships.

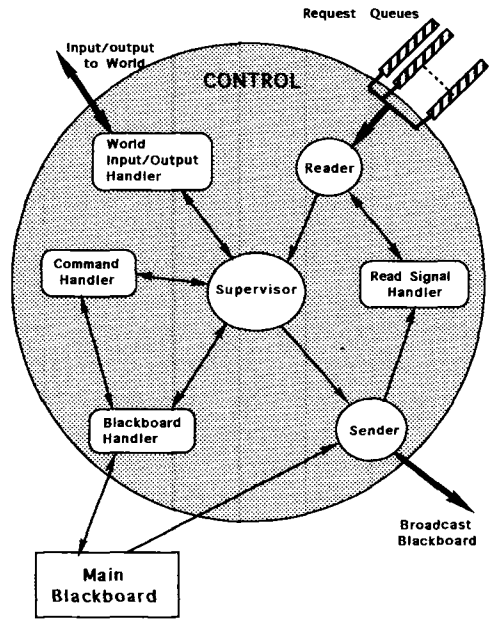

Figure 5. Control

The *Reader* reads incoming requests from intelligent agents. It handles the requests on a first-come first-served basis. The *Reader* has different queues for different kinds of requests such as command or blackboard access requests. The *Blackboard handler* deals with only blackboard related requests and updates the blackboard by adding or deleting rules and facts to/from the blackboard. The *Read signal handler* waits for a read signal from intelligent agents to synchronize the framework. The *Command handler* deals with the system related commands such as clear blackboard or new intelligent agent has entered the framework. The *world input/output handler,* deals with external application programs input/output operations. The *Sender* broadcasts the blackboard, and the *Supervisor* always makes last decision about any request that control has to deal with.

2.4 Blackboard

The blackboard is a data structure that holds information and knowledge that intelligent agents use. The blackboard is organized by the control component. The blackboard has four different members as shown in Figure 6. These members are: 1) *Blackboard name* which holds

the name of the blackboard that is being used, this is necessary if multiple blackboard are in use, 2) *Blackboard status* which is a counter that is incremented after each update of the blackboard, 3) *Private message area* which is used by the control component to send system related messages to the intelligent agents, 4) The *knowledge area* which holds, rules, facts and commands that needs to be broadcast to the intelligent agents.

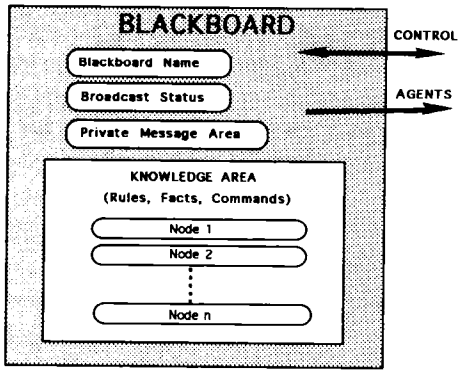

Figure 6. Blackboard

3. A PROTOTYPE IMPLEMENTATION

3.1. Prototype System

Using the SunOS operating system multiprocessing techniques and interprocess communication facilities (IPC), we have developed a prototype system on a Sun platform to demonstrate the concepts described.

In the following sections we provide an overview of the prototype system followed by the discussion of each component of the system. Figure 7 represents the overall architecture of the prototype system. Shared memory has been used to implement the system main blackboard to broadcast messages from control to intelligent agents. Message queues have been used to transfer messages from the intelligent agents to the control. As described in the previous sections, each intelligent agent and control has an input/output port to the world outside of the application framework to interface with the other processes outside of their environment. These outside processes might be any program that uses the application framework. In the prototype system, intelligent agents and control use IPC facilities to interface with the outside programs. [19]

3.2. Control

The control component of the system has been implemented using the C programming language. It runs as a separate process and communicates with the other processes using IPC facilities as described in the previous sections. The control always has to be loaded first. Once the control is loaded, it creates the three incoming control message queues for the requests coming from the intelligent agents and one shared memory to be used as the main system blackboard. The control has read/write access to the main system blackboard and has only read access from the message queues.

3.3. Intelligent Agents

An intelligent agent can be any application, such as an expert system shell, that is able to use IPC facilities on the SunOS operating system. In this prototype system, CLIPS (C Language Production System) [20] was used as intelligent agents. CLIPS is an expert system shell which uses a rule-based knowledge representation scheme. We have extended the CLIPS shell by adding a set of functions to provide the necessary capabilities to use IPC facilities.

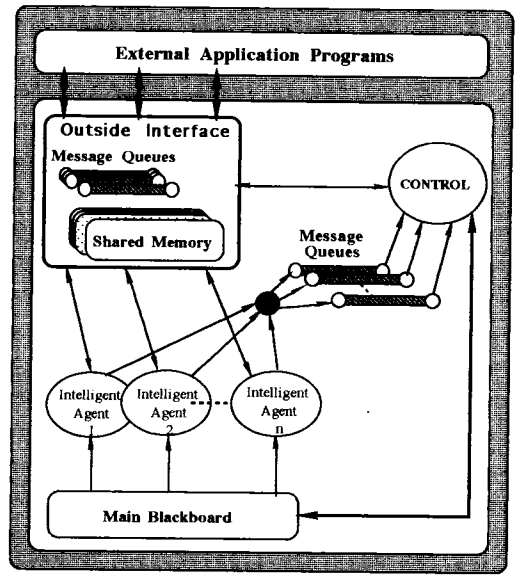

Figure 7. Prototype of Framework

3.4. Fact Transfer Among Intelligent Agents

An intelligent agent can send fact(s) to an individual intelligent agent or to all intelligent agents in the system. Facts stay in the main blackboard until removed by the sender agent or by other agent(s) that has/have the permission to delete the fact.

3.5. Command Transfer Among Intelligent Agents

An intelligent agent can send command(s) to an individual intelligent agent or to all intelligent agents in the system. Commands do not remain on the main blackboard, the receiver agent executes the command immediately upon its arrival. Commands are deleted by the receiver agent(s) as soon as they are executed.

All CLIPS commands are supported by the prototype system. Hence, an intelligent agent can modify the knowledge of other agents via sending the appropriate command. Application programmer should be careful when designing the system since it is possible to remove static knowledge and local facts of the intelligent agent receiving the commands.

3.6 Rule Transfer Among Intelligent Agents

An intelligent agent can send rule(s) to an individual intelligent agent or to all intelligent agents in the system. Rules stay on the main blackboard until removed by the sender agent or other agent(s) that has the right permission to delete the rule. CLIPS format should be followed to represent rules.

3.7. Knowledge Transfer Among Intelligent Agents

Knowledge can be exchanged among intelligent agents by using combination of facts, rules and commands transfers. Different methodologies can be used for knowledge transfer; knowledge can be exchanged among intelligent agents in temporary or permanent bases.

4. CONCLUSIONS

The focus of this research is dynamic knowledge exchange among intelligent agents. By introducing simple communication protocols among intelligent agents, we have introduced a framework through which intelligent agents can exchange knowledge in a dynamic environment. Using the proposed framework common knowledge can be maintained by one intelligent agent and broadcasted to the other agents when necessary.

Research related to the proposed system architecture has been limited to the exchange of simple facts and goals among intelligent agents. However, in order to successfully respond to unexpected events in dynamic environments, it is imperative to have the capability of dynamic knowledge exchange among intelligent agents.

The framework proposed in this research extends the current technology by allowing dynamic knowledge exchange among intelligent agents. In a dynamic environment, intelligent agents must be responsive to unanticipated conditions. When such conditions occur, an agent may be required to terminate previously planned and scheduled courses of action, and replan, reschedule, start new activities, and initiate a new problem solving process, in order to successfully respond to the new conditions. Problems occur when an intelligent agent does not have sufficient knowledge to properly respond to the new condition. The framework presented in this research would allow an intelligent agent to react to any knowledge deficiency. Using the proposed framework, new knowledge can be transferred when an intelligent agent is unable to solve the problem due to its limited knowledge. Once the knowledge has been transferred, the intelligent agent can either retain the transferred knowledge permanently or remove this knowledge after the problem has been resolved.

Complex systems can be built using different networking techniques. The proposed framework is based on the blackboard model of distributed problem solving. As presented in the literature review, current systems based on a blackboard architecture only allow facts to be posted or removed from the system main blackboard in a distributed problem solving environment. In this research, the basic blackboard architecture has been extended to allow transfer of knowledge (i.e., rules) as well as facts, among the intelligent agents.

We believe that dynamic knowledge exchange would be an important feature for any application in which unanticipated conditions or events occur. Using the proposed dynamic knowledge exchange capability, cooperative problem solving sessions can be initiated where each agent can share its problem relevant knowledge with other agents to resolve the problem. An obvious advantage of this capability is the elimination of redundant knowledge and hence the improved utilization of the system memory capacity. In addition, by using this framework a form of learning can take place and thus additional problem solving knowledge is created.

A prototype system has been developed an a Sun workstation platform in order to demonstrate the operational aspects of the proposed framework. Interprocess Communication Facilities (IPC) from the SunOS operating system have been utilized to develop the system blackboard as well as to implement the communication protocols which allow interaction among intelligent agents.

The CLIPS expert system shell has been used to represent intelligent agent(s) in the prototype system. The functionality of CLIPS has been extended to support IPC facilities as well as the proposed framework communication protocols. In addition, new functions have been added to the CLIPS environment to allow for dynamic insertion and/or deletion of rules and facts into knowledge and fact bases.

Other possible areas of application for the proposed framework include: Distributed Intelligent Simulation (DIS), Robotics and Manufacturing (RM) and Intelligent Simulation and Training (IST). In DIS applications, each object in the simulation can be represented by an intelligent agent. In this case an agent would encapsulate the knowledge and behavior representing the real-world entity in the domain. In RM applications, functions such as vision and assembly/disassembly tasks can benefit from the dynamic knowledge exchange capability to identify new objects or perform new tasks without interruption. In IST applications, training simulators have been networked using protocols such as Distributed Interactive Simulation to provide realism to the training function. In a war game simulation exercise, entities such as officers and soldiers can be modelled as intelligent agents. In this case, an intelligent agent representing an officer can dynamically transfer appropriate knowledge to each soldier according to each soldier's specific condition at the time.

REFERENCES

[1] Dodhiawala, R. T., Sridharan, N. S. and Pickering C. *A Real-Time Blackboard Architecture*, Blackboard Architectures and Applications. Academic Press, San Diego, CA, 1989.

[2] Golstein, G. *PRAIS: Distributed, Real-Time Knowledge_Based Systems Made Eassy*. Proceeding of the First CLIPS User's Group Conference, Houston, TX, 1990.

[3] Myers, L. Johnson, C and Johnson, D. *MARBLE: A Systems for Executing Expert System in Parallel.* Proceeding of the First CLIPS User's Group Conference, Houston, TX, 1990.

[4] Schultz, R. D. and Stobie, I. C. *Building Distributed Rule-Based Systems Using the AI Bus.* Proceeding of the First CLIPS User's Group Conference, Houston, TX, 1990.

[5] Raulefs, P. *Toward a Blackboard Architecture for Real-Time Interactions with Dynamic Systems.* Blackboard Architectures and Applications. Academic Press, San Diego, CA, 1989.

[6] Corkill, D. D., Gallagher K. Q, and murray, K. E. *GBB: A Generic Blackboard Development System,* Blackboard Systems. Addison Wesley, Reading, Mass, 1988.

[7] Gallagher K. Q. and Corkill, D. D. *Performance Aspects of GBB*, Blackboard Architectures and Applications. Academic Press, San Diego, CA, 1989.

[8] Gilmore, J. F., Roth, S. P. and Tynor S. D. *A Blackboard System for Distributed Problem Solving.* Blackboard Architectures and Applications. Academic Press, San Diego, CA, 1989.

[9] Taylor, J. and Myers, L. *Executing CLIPS Expert Systems in a Distributed Environment.* Proceeding of the First CLIPS User's Group Conference, Houston, TX, 1990.

[10] Gasser, L. Braganza, C. and Herman, N. *MACE : A Flexible Testbed for Distributed AI Research.* Distributed Artificial Intelligence, Morgon Kaufman Publisher, 1987.

[11] Rice, J., Aiello, N., and Nii, H. P. *See How They Run... The Architecture and Performance of Two Concurrent Blackboard Systems.* Blackboard Systems. Addison Wesley, Reading, Mass, 1988.

[12] Nii, H. P. *Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures* , AI Magazine, Summer 1986, pp. 38-53.

[13] Nii, H. P. *Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective.* AI Magazine, August 1986, pp. 82-106.

[14] Erman, L. D., Hayes-Roth, F., Lesser, R. V. and Reddy, D. R. *The Hearsay-II Speech-Understanding System : Integrating Knowledge to Resolve Uncertanity.* Blackboard Systems. Addison Wesley, Reading, Mass, 1988.

[15] Craig, I. D. The Cassandra Architecture: distributed control in a blackboard system. Ellis Horwood Limited, West Sussex, 1989.

[16] Findler, N. V. *Distributed Control of Collaborating and learning Expert Systems for Street Traffic Signals,* IFAC Symposium on Distributed Intelligence Systems, Arlington, VA, 1991.

[17] Genesereth, M.R., Fikes, R. E. *Knowledge Interchange Format Reference Menual,* Stanford University Logic Group, 1992.

[18] Genesereth, M.R., Ketchpel, S. P., *Software Agents,* ACM Communication, July 1994, pp. 48-53.

[19] Cengeloglu, Y., Sidani, T. and Sidani, A. *Inter/Intra Communication in Intelligent Simulation and Training Systems (ISTS).* 14th Conference on Computers and Industrial Engineering, Cocoa Beach, March 1992.

[20] CLIPS Version 5.1 User's Guide, NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, TX, 1991