Cooperative Agent-based Software Architecture for Distributed Simulation

V.K.Murthy[1] , E.V.Krishnamurthy[2]

***Abstract*- This paper proposes a cooperative multi-agent model using distributed object-based systems for supporting distributed virtual environment and distributed simulation technologies for military and government applications. The agent model will use the condition-event driven rule based system as the basis for representing knowledge. In this model, the updates and revision of beliefs of agents corresponds to modifying the knowledge base. These agents are reactive and respond to stimulus as well as the environment in which they are embedded. Further, these agents are smart and can learn from their actions. The distributed agent-based software architecture will enable us to realise human behaviour model environment and computer-generated forces (also called computer-generated actor (CGA)) architectures. The design of the cooperative agent-based architecture will be based on mobile agents, interactive distributed computing models, and advanced logical modes of programming. This cooperative architecture will be developed using Java based tools and distributed databases.**

***Index Terms*- Agent-based simulation, Computer Generated Forces, Intelligent Agents, Distributed Simulation, Agent-based Combat Modeling**

I. INTRODUCTION

A cooperative multi-agent model is proposed using object-based systems for supporting distributed virtual environment and distributed simulation technologies for military and government applications. In this paper, we propose a distributed agent-based software architecture that will enable us to realise intelligent military operations planning systems and computer-generated forces (also called computer-generated actor (CGA)) systems. These CGA systems will be useful in modeling cyber warfare, or information warfare.

1 Colleges of Applied Sciences, Ministry of Higher Education, Sultanate of Oman, and Australian National University, Canberra, ACT 0200, Australia edayathuk@gmail.com and ariyalurk@gmail.com.
2Australian National University, Canberra, ACT 0200 Australia, Evk.Krishnamurthy@anu.edu.au

Agents have intentions and actions. They are autonomous and they have a built in control to act only if they want to. In addition, agents are flexible, proactive and have multithreaded control. An agent is a system that is capable of perceiving events in its environment, or representing information about the current state of affairs and of acting in its environment guided by perceptions and stored information. Agents can be classified based on their functionality: collaborative agents that cooperate; interface agents that act as personal assistants; mobile agents that migrate among hosts to enhance the efficiency of computation and improve the network throughput; information agents that manage, manipulate and collate information from many distributed sources; reactive agents that respond to stimulus and respond in an environment where they are embedded; smart agents that learn from their actions; hybrid agents that can combine any of the functionalities of the above agents.

We describe in detail how a set of smart agents can be used for collaboration and cooperation. We will develop an integrated agent-based model that links with the distributed software engineering methodology. This model also provides an insight into the self-organized criticality in a network of agents [1-9]. These smart cooperative software agents are useful in realizing distributed military operations planning systems. The multi-agent model can simulate the collaboration and cooperation of human participants and hence we can construct a suitable human behaviour model in a distributed environment.

The agent model has the following features [10-12]:

1. Derivation of events and actions by interpreting inputs. A rule-based system provides the basis for representing knowledge. Update and revision of beliefs are formalized.
2. Process of going from goals to plans and actions provides for program design.
3. Deterministic, nondeterministic and probabilistic choice functions can be incorporated. Hence cooperation and collaboration among agents are possible.
4. Condition + events, intention + action, subjunctive/ abductive reasoning and failure recovery can be embedded in the transactional and workflow approach.
5. Algorithm and protocol design for distributed systems based on utility functions becomes easy.

6. Emergence in which the total system exhibits new properties can be realised in a large number of interconnected agents.
7. Permits development of distributed software tools using Java for multi-agent systems engineering.
8. Has the simplicity and adaptability for realisation as a distributed transaction-based paradigm for collaboration.
9. Provides an insight into the self-organized criticality in a network of agents.

II. AGENT MODEL

A multi-agent system consists of the following subsystems [6]:

(1) Environment U: Those states which completely describe the universe containing all the agents.

(2) Input (Percept): This is an input from the environment**.** Depending upon the sensory capabilities (input interface to the universe or environment) an agent can partition U into a standard set of messages T, using a sensory function Perception (PERCEPT):

PERCEPT: U □ T.

PERCEPT is interpreted by an agent and involves various senses such as see, read and hear. The messages are assumed to be of standard types based on an interaction language that is interpreted identically by all agents.

(3) Mind M: The agent has a mind M (namely, a problem domain knowledge consisting of an internal database for the problem domain data and a set of problem domain rules) that can be clearly understood by the agent without involving any sensory function. The database D sentences are in first order predicate calculus (also known as extensional database) and agents mental actions are viewed as inferences arising from the associated rules that result in an intentional database, that changes (revises or updates) D.

The beliefs are first order logic sentences resulting from information about the environment at a certain time. These beliefs can be of three types:

(i)Elementary belief: This is assumed or self supported,

(ii) Derived belief: This is got from perception and communication.

(iii) Inferential belief: This is got through analysis.

An agent's mind therefore knows what the belief is, how it was arrived at and why it is true. A distributed belief is composed of the union of the beliefs of all agents. Thus M can be represented by an ordered pair of elements (D, P). D is a set of beliefs about objects, their attributes and relationships stored as an internal database and P is a set of rules expressed as preconditions and consequences (conditions and actions). When T is input, if the conditions given in the left-hand side of P match T the elements from D that correspond to the right-hand side are taken from D and suitable actions are carried out locally (in M) as well as on the environment.

(4) Organizational Knowledge (O): Since each agent needs to communicate with the external world or other agents, we assume that O contains all the information about the relationships among the different agents. For example, the connectivity relationship for communication, the data dependencies between agents, interference among agents with respect to rules, information about the location of different domain rules are in O.

(5) INTRAN: M is suitably revised or updated by the function called Internal transaction (INTRAN). Revision means acquisition of new information about the world state, while update means change of the agent's view of the world. Revision of M corresponds to a transformation of U due to occurrence of events and transforming an agent's view due to acquisition of new information that modifies rules in P or their mode of application (deterministic, nondeterministic or probabilistic) and corresponding changes in database D (e.g modifying the tax-rules). Updates to M correspond to changes in U due to the occurrence of events that changes D but not P (e.g. inserting a new tax -payer in D). That is:

INTRAN: M X T □ M

(6) EXTRAN: External action is defined through a function called global or external transaction (EXTRAN) that maps an epistemic state and a partition from an external state into an action performed by the agent. That is:

EXTRAN: M X T Æ A

This means that the current state of mind and a new input activates an external action from A.

(7) EFFECT: The agent also has an effectory capability on U by performing an action from a set of actions A (ask, tell, hear, read, write, speak, send, smell, taste, receive, silent), or more complex actions. Such actions are carried out according to a particular agent's role and governed by an etiquette called protocols. The effect of these actions is defined by a function EFFECT, that modifies the world states, through the actions of an agent:

EFFECT: A X U □ U; EFFECT can involve additions, deletions and modifications to U. Thus an agent is defined by:

(U, T, M (P,D),O,A,PERCEPT,INTRAN,EXTRAN, EFFECT).

The interpreter repeatedly executes selected rules in P, until no rule can be fired.

The nature of internal production rules P, their mode of application and the action set A determines whether an agent is deterministic, nondeterministic, probabilistic or fuzzy. Rule application policy in a production system P can be modified by:

(1) Assigning probability/fuzziness for applying a rule.

(2) Assigning strength to a rule by using a measure of its past success.

(3) Introducing a support for a rule by using a measure of its likely relevance to the current situation.

The above factors provide for competition and cooperation among the different rules. Such a model is useful for collaboration and cooperation involving many agents.

III. AGENT-BASED COLLABORATION

Agent-based collaboration is an interactive process among many smart mobile agents that results in varying degrees of cooperation and competition and ultimately leads to commitment [3, 4, 5]. This will result in total agreement, consensus or disagreement. Agents connected by a network sharing a common knowledge base exchange private knowledge through transactions and create new knowledge. Each agent transacts its valuable private knowledge with other agents and the resulting transactional knowledge is shared as common knowledge. Agents may benefit by

exchanging their private knowledge if their utility will be increased. This knowledge is traded in if their utilities can be improved. If during a transaction the difference between external and internal knowledge is positive this difference is added to private knowledge; else it is treated as common knowledge. The graph that links the agents is called the collaboration graph. A collaboration protocol is viewed as a set of public rules that dictate the conduct of an agent with other agents to achieve a desired final outcome in sharing the knowledge and performing actions that satisfy a desired goal satisfying some utility functions. A directed graph can be used to represent a collaboration process. This graph expresses the connectivity relationship among the agents, that can be real or conceptual and can be dynamic or static depending upon the problem at hand.

Multi-agents can cooperate to achieve a common goal to complete a task to aid the customer. The negotiation follows rule-based strategies that are computed locally by its host server. Here competing offers are to be considered; occasionally cooperation may be required. Special rules may be needed to take care of risk factors, domain knowledge dependencies between attributes, positive and negative end conditions. When making a transaction several agents have to negotiate and converge to some final set of values that satisfies their common goal. Such a goal should also be cost effective so that it is in an agreed state at the minimum cost or a utility function. To choose an optimal strategy each agent must build a plan of action and communicate with other agents. For communication among the agents one can think of various models: (i) arbitration model in which each client-agent communicates through an arbitrator (ii) auction in which there is a central coordinator who collects the information from participants and make them public, and (iii) Direct search which involves catalogue/directory service. When there is no coordinator, collaboration and negotiation can lead to a self-organized criticality and this can lead to a speculation bubble or a crash, or a stagnation and a phase transition among such states.

IV. AGENT-BASED NEGOTIATION

Human problem solving uses an act-verify strategy through preconditions and actions. When a human solves a problem, the solution process has a similarity to the transaction handling problem; for each transaction is an exploratory non pre-programmed real-time procedure that uses a memory recall (Read), acquires a new information and performs a memory revision (Write). Each transaction is also in addition provided with the facility for repair (recovery-Undo) much like the repair process encountered in human problem solving. In human problem solving, several independent or dependent information is acquired from various knowledge sources and their consistency is verified before completing a step of the solution to achieve each subgoal; this process corresponds to committing a subtransaction in a distributed transaction processing system, before proceeding to reach the next level of subgoal arranged in a hierarchy. Thus the transactional approach provides for a propose, act and verify strategy by offering a nonprocedural style of programming (called 'subjunctive programming') in which a hypothetical proposal or action (what if changes) is followed by verification, commitment or abort and restoration. So this paradigm is well-suited for smart agent-based negotiation in distributed simulation.

V. DISTRIBUTED NEGOTIATION

A distributed negotiation protocol has the following properties:

(1) The negotiation leads to a finite number of states.

(2) The negotiation process does not enter cyclic or infinite sequences but always reaches a terminal state.

We now describe how to carry out distributed multi-agent negotiation by sending, receiving, handshaking and acknowledging messages and performing some local computations. A multi-agent negotiation has the following features:

1. There is a seeding agent who initiates the negotiation.
2. Each agent can be active or inactive.
3. Initially all agents are inactive except for a specified seeding agent, which initiates the computation.
4. An active agent can do local computation, send and receive messages and can spontaneously become inactive.
5. An inactive agent becomes active, if and only if, it receives a message.
6. Each agent may retain its current belief, revise or update its belief as a result of receiving a new message by performing a local computation. If it modifies its belief, it communicates its new belief to other concerned agents; else it does not modify its belief and remains silent.

VI. NEGOTIATION TERMINATION

In order that the distributed negotiation protocol is successful we need to ensure that the negotiation process ultimately terminates. For this purpose, we now describe an algorithm that can detect the global termination of a negotiation protocol. Let us assume that the N agents are connected through a communication network represented by a directed graph G with N nodes and M directed arcs. Let us also denote the outdegree of each node i by Oud (i) and indegree by Ind(i). Also we assume that an initiator or a seeding agent exists to initiate the transactions. The seeding agent (SA) holds an initial amount of money C. When the SA sends a data message to other agents, it pays a commission:

C/(Oud (SA) + 1) to each of its agents and retains the same amount for itself. When an agent receives a credit it does the following:

a. Let agent j receive a credit C(M(i)) due to some data message M(i) sent from agent i . If j passes on data messages to other agents j retains C((M(i)) / (Oud(j)+1) for its credit and distributes the remaining amount to other Oud(j) agents. If there is no data message from agent j to others, then j credits C(M(i)) for that message in its own savings account; but this savings will not be passed on to any other agent, even if some other message is received eventually from another agent.

b. When no messages are received and no messages are sent out by every agent, it waits for a time-out and sends or

broadcasts or writes on a transactional blackboard its savings account balance to the initiator.
c. The initiator on receiving the message broadcast adds up all the agents' savings account and its own and verifies whether the total tallies to C.
d. In order to store savings and transmit commission we use an ordered pair of integers to denote a rational number and assume that each agent has a provision to handle exact rational arithmetic. If we assume C=1, we only need to carry out multiplication and store the denominator of the rational number.

We prove the following theorems to describe the validity of

the above algorithm:

Theorem 1: If there are negotiation cycles that correspond to indefinite arguments among the agents (including the initiator itself) then the initiator cannot tally its sum to C.

Proof: Assume that there are two agents i and j are engaged in a rule dependent argument cycle. This means i and j are revising their beliefs forever without coming to an agreement, and wasting the common resource C. Let the initial credit of i be x. If i passes a message to j, then i holds x/2 and j gets x/2. If eventually j passes a message to i ,then its credit is x/4 and i has a credit x.3/4 ; if there is continuous exchange of messages for ever then their total credit remains (x - x/2k) with x/2k being carried away by the message at k th exchange. Hence the total sum will never tally in a finite time.

Theorem 2: The above algorithm terminates if and only if the initiator tallies the sum of all the agents savings to C.

Proof: If part: If the initiator tallies the sum to C this implies that all the agents have sent their savings and no message is in transit carrying some credit and there is no chattering among agents.

Only if part: The credit assigned can be only distributed in the following manner:

a. An agent has received a message and credit in a buffer; if it has sent a message then a part of the credit is lost; else it holds the credit in savings.

b. Each message carries a credit; so, if a message is lost in transit or communication fails then total credit cannot be recovered.

Thus termination can happen only if the total sum tallies to C, i.e., the common resource is not wasted and all the agents have reached an agreement on their beliefs.

We will illustrate the above algorithm using the E-auction scenario in which there is a single auctioneer and a set of clients participating in the auction.

VII. NEGOTIATION EXAMPLE

Auction process is a kind of controlled competition among a set of agents (clients and auctioneer) coordinated by the auctioneer. We use this example since it is simple to explain the concept of agents, their beliefs and actions as outlined in Section 2. In this example, the belief is first obtained from the auctioneer and other clients through communication and these are successively updated. Finally, the distributed belief among all participants is composed of all the existing beliefs of every agent involved in the process.

The rules that govern the auction protocol are as follows:

(1) At the initial step the auctioneer-agent begins the process and opens the auction.
(2) At every step, decided by a time stamp, only one of the client-agent is permitted to bid and the auctioneer relays this information. The bidding client agent is called active and it does not bid more than once and this client becomes inactive until a new round begins.
(3) After the auctioneer relays the information a new client becomes active and bids a value strictly greater than a finite fixed amount of the earlier bid.
(4) When at a given time-out period no client-agent responds, the last bid is chosen for the sale of the goods and the auction is closed.

Note that the Rule 3 here corresponds to English auction, but it can be different depending upon the nature of the auction.

Agent-Based Protocol

Let us assume that there are three clients (A, B, C) and an auctioneer G. The auctioneer G initiates the auction. Then each of the clients A,B and C broadcasts their bid and negotiates, and the auctioneer relays the information. The bidding value is known to all the clients and the auctioneer. When the bid reaches a price above a certain reserve price, and no bid comes forth until a time-out, G terminates the auction and the object goes under the hammer for that price. In E-auction the above scenario can be realised and the negotiation termination algorithm can be used.

At initiation, the node G is the seeding agent (auctioneer). It transmits the information to each client the beginning of the E-auction. Also it starts with a credit 1 and retains a credit of 1 /(Oud (SA)+ 1 to itself, and transmits the same amount to its neighbours (A, B, C) which in this case is 1/4. The retained credit for each transmission is indicated near the node. To start with the agent-client A bids a value. Then all clients and G get this information and the credits. Then agent-client node B updates its earlier belief from the new message received from G; but the other nodes A, C do not update their initial beliefs and remain silent .The agent-client node C then bids. Finally as indicated in the rules described above, we sum over all the retained credits after each transmission.

VIII . AGENT-BASED SIMULATION

The agent negotiation system can be used to model a distributed battlefield simulation system. In this system, the military operations are modelled as a distributed process among many soldiers (agents) coordinated by the group commander (controlling agent).

In this simulation system, the domain data D, rules P and organizational knowledge O are based on three factors:
(1) The experience and knowledge of a soldier is based totally on his criteria (elementary belief)
(2) The soldier acquires knowledge through communication with other other soldiers and commanders; such a soldier is called a fundamentalist (derived belief).
(3) The soldier acquires knowledge by observing the behavior of other soldiers and commanders; such a soldier is called a trend chaser (inferential belief). In practice a soldier is influenced by the above factors and the modified knowledge is incorporated in D, P and O.

In a battlefield (military) simulation system, a soldier or a commander (an agent) can perform various actions including Attack and Retreat. Each agent can communicate with other agents and this creates a bond among them. This results in the modification of the organizational knowledge O. This bond is created with a certain probability determined by a parameter, which characterises the willingness of a soldier or commander to comply with others.

We can assume that any two soldiers or commanders (agents) are randomly connected with a certain probability. This divides the agents into clusters of different sizes whose members are linked either directly or indirectly via a chain of intermediate agents. These groups are coalitions of military participants who share the same opinion about their activity. The decision of each group is independent of its size and the decision taken by other clusters.

Using percolation theory [11] it can be shown that when every agent is on average connected to another, more and more agents join the spanning cluster, and the cluster begins to dominate the overall behaviour of the system. This gives rise to "Offensive Action" (if all the soldiers and commanders decide to attack) and a "Defensive Action" (if all the soldiers and commanders decide to retreat). Accordingly, an analogy exists between "Offensive Action" or "Defensive Action" and critical phenomena or phase transitions in physics. Thus a distributed agent system can eventually enter into a phase-transition like situation [1, 2, 8, 9, 11].

When soldiers and commanders (agents) collaborate in a battlefield simulation system, the collaboration graph consists of many nodes and edges. As more military participants join and their collaboration increase, the number of links increase and the collaboration graph grows. The links among the agents can be established in a certain preferential manner rather than a uniform distribution. Recently, Barabasi and Albert [2] observe that the growth and preferential attachment leads to a power-law distribution, namely, the probability P(k) that each agent has k links is k - x, where x = 2.3. Thus the development of distributed complex systems is governed by robust self-organizing phenomena that go beyond the particulars of the individual systems. Therefore complex systems involving a large number of agents will self organize into a scale-free state. This phenomenon will be useful in complex distributed military operations planning systems involving many smart agents.

IX. CONCLUSION

We have proposed a cooperative multi-agent model using distributed object-based systems for supporting distributed simulation technologies for military applications. These agents are reactive and respond to stimulus as well as the environment in which they are embedded. Further, these agents are smart and can learn from their actions. The distributed collaborative agent-based software architecture will enable us to realise human behaviour model environment and computer-generated forces architectures. This model also provides an insight into the self-organized criticality in a network of agents.

REFERENCES

[1] Bak, P. 1996. *How Nature works: The Science of Self-organized criticality*. Springer, New York.

[2] Barabasi, A and R. Albert. 1999. "Emergence of scaling in random networks." *Science 286*, 509-512.

[3] Chen, Q. and U. Dayal. 2000. "Multi agent cooperative transactions for E-commerce." *Lecture Notes in Computer Science 1901*, 311-322.

[4] DeLoach, S. A. 2001. "Multiagent Systems Engineering." *International Journal of Software Engineering and Knowledge Engineering 11*, 231-258.

[5] Dignum, F. and C. Sierra. 2000. "Agent Mediated E-Commerce." *Lecture Notes in Artificial Intelligence 2003*.

[6] Fisher, M. 1995. "Representing and executing agent-based systems." *in Intelligent Agents*,Woolridge,M.,and Jennings, N.R (Eds.), Lecture Notes in Computer Science,Vol.890, Springer-Verlag, New York, pp. 307-323,1995.

[7] N.R.Jennings,On agent -based software engineering, *Artificial Intelligence*, Vol.117, pp.277-296,2000.

[8] S.Kirkpatrick and .B. Selman, Critical behaviour in the satisfiability of random boolean expressions, *Science*, Vol.264, pp.297-1301,1994.

[9] A.Lloyd and R.M.May, How viruses spread among computer and people, *Science*, Vol.292, pp.1316-1317.,2001.

[10] Nagi, K. 2001. *Transactional agents*, Lecture Notes in Computer Science. New York, Springer Verlag.

[11] Paul, W. and J. Baschnagel. 2000. *Stochastic Processes*. Springer Verlag, New York.

[12] M.Winikoff et al, *Simplifying the development of intelligent agents*, Lecture Notes in Artificial Intelligence, Vol. 2256, pp. 557-56,2001.