Defence Science Journal, Vol. 53, No. 3, July 2003, pp. 267-273
© 2003, DESIDOC

Agent-oriented Programming in Defence Domain

Sunil Doiphode
Research & Development Establishment (Engineers), Pune - 411 015

ABSTRACT

Research in distributed artificial intelligence has given rise to agent-oriented programming (AOP), an advanced software modelling paradigm. It has several benefits when compared with the existing development approaches, in particular, the ability to let agents represent high-level abstractions of active entities in a software system. Although still young and under evolution, this paradigm has already shown particular promise in a number of areas. This paper gives an overview of this paradigm, its benefits over the other conventional programming paradigms being used. It also proposes the decision support system model for the military domain.

In the proposed system there are certain critical issues, which need to be focused upon. The existing conventional paradigms are inadequate to deal with these issues. This paper identifies these critical issues and discusses how AOP can address these issues.

Keywords: Agent-oriented programming , distributed artificial intelligence, artificial intelligence, decision support system, object-oriented programming, JAVA, intelligent agents, mobile agents, BDI agent

1. INTRODUCTION

The agent-oriented programming (AOP) paradigm is one of the evolving programming paradigms. The conventional programming paradigms like structural programming, object-oriented programming (OOP), logical programming, etc. were inadequate in some of the application areas. The OOP has been used for building intelligent agents, with the limitation that it cannot represent complex mental attitudes. With logical programming, it is possible to represent and infer relationships among mental attitudes, such as intentions, goals, and beliefs, with limitations in the usage of capabilities of action.

In the Defence domain, certain applications do not perform up to the mark for various aspects. Sometimes, this is not due to the poor design or programming, but due to the incapability of the underlined paradigm used for software modelling.

2. AGENT-ORIENTED PROGRAMMING

The AOP is seen as an improvement and extension of the OOP. The OOP, on the other hand, can be seen as a successor of structured programming[1]. The term AOP was introduced by Shoham[2], in 1993. In OOP, the main entity is the object. An object is a logical combination of data structures and their corresponding methods (functions). Objects are successfully being used as abstractions for passive entities in the real world, and agents are regarded as possible successors of objects since these can improve the abstraction of

Revised 16 October 2002

active entities. Agents are similar to objects, but they also support structures for representing mental components (for instance, beliefs, goals, actions, and plans). Another important difference between the AOP and the OOP is that objects are controlled from the outside (white box control), as opposed to agents that have autonomous behaviour, which is not directly controllable from the outside (black box control).

The OOP has the limitation that it cannot represent complex mental attitudes. Logical programming can represent mental attitudes. It has limitations in the usage of capabilities of action, which can be represented in the OOP. Therefore, it can be presumed that the OOP and the logical programming paradigms merge to form the AOP, in the process of evolution. The process of the evolution of programming paradigms can be expressed diagrammatically (Fig. 1).

There are, however, various, if not contradictory, definitions and architectures of an intelligent agent. As per general consensus, it is a type of software that shows some degree of autonomy and social ability, and combines proactive and reactive behaviours[5]. One of these architectures, which is better known and easy to understand, is belief-desire-intention (BDI) agent architecture.

2.1 Belief-desire-intention Agent

The agent, as described here, is an autonomous piece of software, which has explicit goals or desires to achieve, and is preprogrammed with plans or behaviours to achieve these goals under varying circumstances. Set to work, the agent pursues given goals, adopting the appropriate plans, or intentions, according to its current beliefs about the state of the world, so as to perform the intended role. Such an intelligent agent is generally referred to as a belief-desire-intention (BDI) agent (Fig. 2).

3. PROPOSED MODEL OF DECISION SUPPORT SYSTEM FOR DEFENCE

The proposed model increases the coordination and cooperation between all the constituents of the Defence, which are necessary to take any decision (Fig. 3). It automates certain defence actions. Some commands passed on in this model are automatic in nature and some are suggestive in nature. For training purpose, in this model, the other human beings can be simulated to agents. The proposed model includes the following components:

3.1 Defence Forces

The Defence Forces have interoperating agents (Ad1, Ad2, Ad3) as well as intraoperating agents (Ar1, Ar2,...., Arn for the Army)(Af1, Af2,...., Afk for the Air Force) (Av1, Av2,... ., Avl for the Navy). If there are n entities in a force, then in an ideal case, the intraoperating agents working in that force will be $n(n-1)/2$. These agents exhibit the social behaviour of team working. Here, the data collection can be through various sensors and human interaction. The data dissipation can be in terms of commands, opinions/suggestions for automated actions or the human actions. All the forces communicate among each other either by the agents like Ad1, Ad2, and Ad3 or through coordinator agent. The model includes the following components for the Defence Forces:

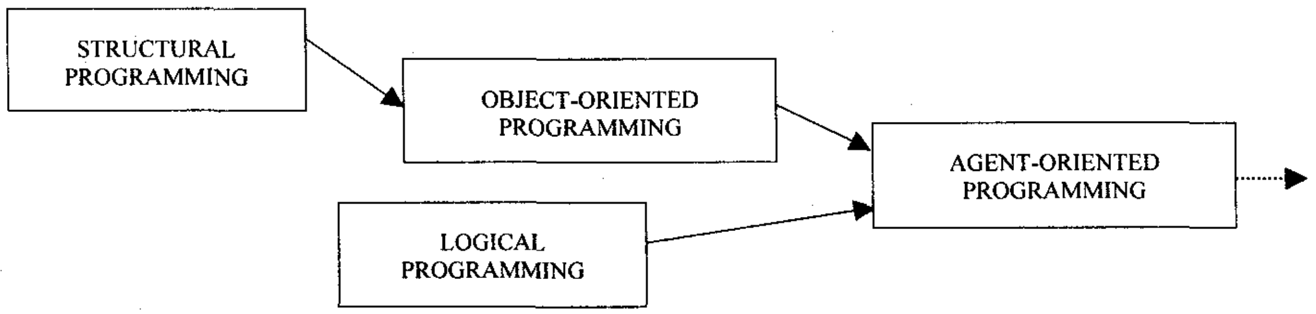


Figure 1. Programming paradigm evolution

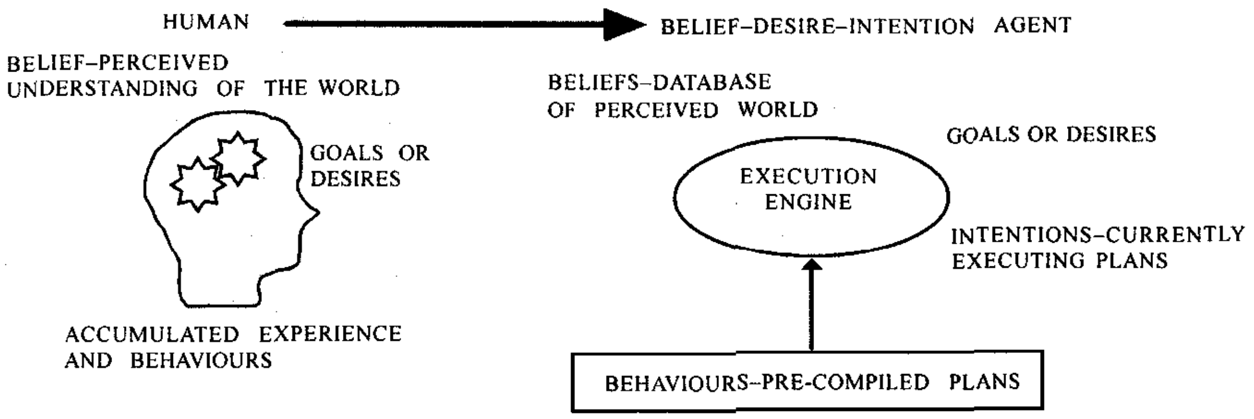

HUMAN
BELIEF–DESIRE–INTENTION AGENT
BELIEF–PERCEIVED UNDERSTANDING OF THE WORLD
GOALS OR DESIRES
ACCUMULATED EXPERIENCE AND BEHAVIOURS
BELIEFS–DATABASE OF PERCEIVED WORLD
GOALS OR DESIRES
EXECUTION ENGINE
INTENTIONS–CURRENTLY EXECUTING PLANS
BEHAVIOURS–PRE-COMPILED PLANS

Figure 2. An agent–intentional or belief-desire-intention agent

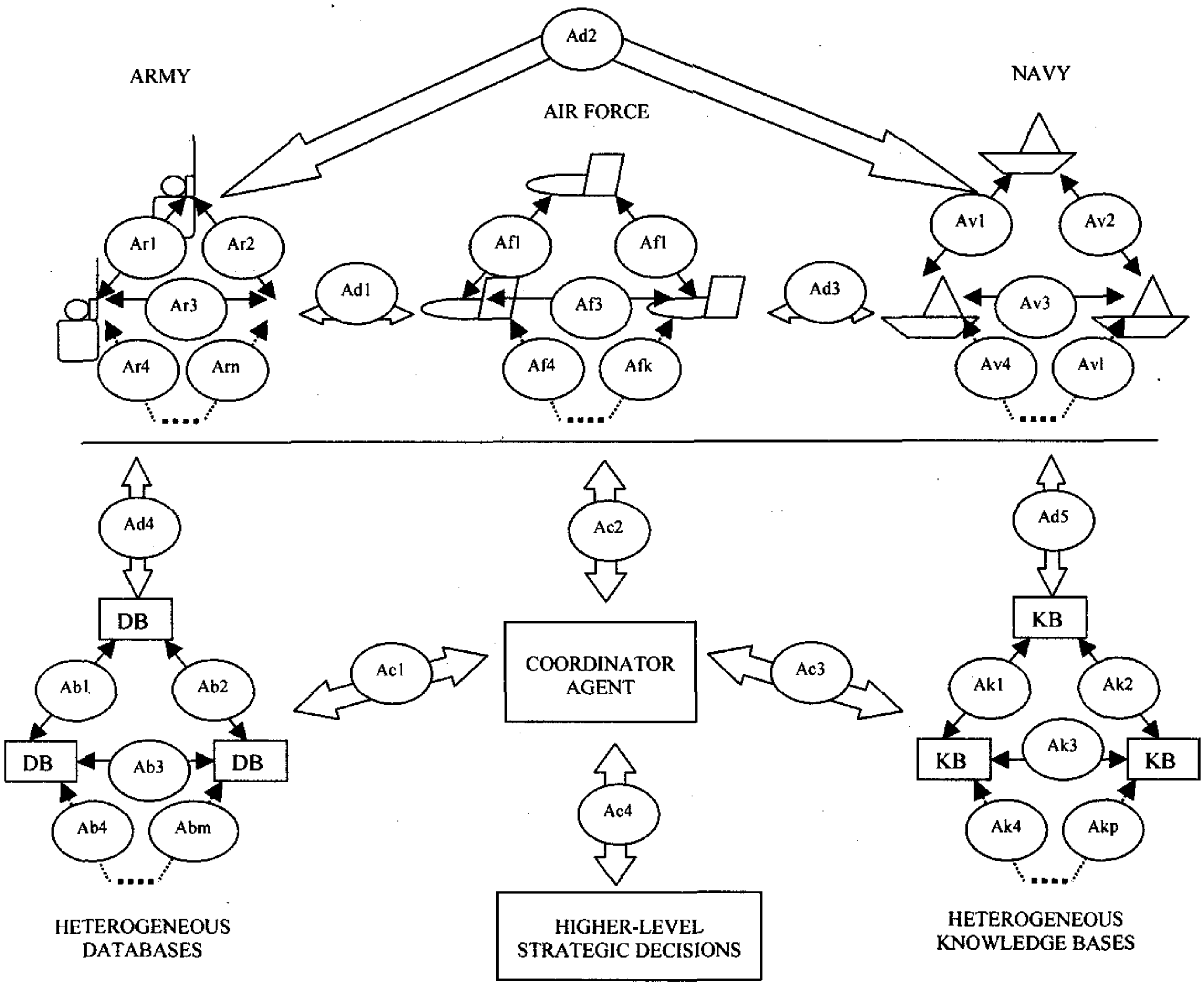


Figure 3. Proposed model of decision support system for Defence

- Army
- Air Force
- Navy

3.2 Data

The source of data is not only agents but also the other applications, which may not be agent-oriented. The model includes the following components for data storage:

- Heterogeneous databases
- Heterogeneous knowledge bases

The data in this model is stored in the different databases and knowledge bases, which are geographically scattered and heterogeneous in nature. The Defence Forces may have their own different data (databases and knowledge bases). Databases and knowledge bases are made logically one entity using the agents Ab1,Ab2..Abm and Ak1,Ak2,.. Akp respectively. All the forces interact with these logical entities through agents Ad4 and Ad5 or through coordinator agent.

3.3 Coordinator Agent

The main role of this agent, as its name suggests, is to coordinate communications among the Defence Forces, databases, knowledge bases, and the higher-level executives who take the strategic decisions. This agent, for this purpose uses the other agents Ac1, Ac2, Ac3, and Ac4.

3.4 Higher-level Strategic Decisions

Higher-level executive authority considers all the factors projected by this model and the other factors, which are not taken care of by the system, and the strategic decisions are communicated to the system through the agent Ac4.

4. ISSUES ADDRESSED BY AOP IN THE PROPOSED MODEL

4.1 Simulation of Human Behaviour & Team Behaviour

In the past, computer simulation has been used in the Defence domain to support procurement, force development, evaluation of C^3 (communicate, command, and control) structures and for training. But, modelling and simulation became complex as multirole, multi-platform, and multisystem aspects are considered. The complexity of this task is further increased by the difficulty in modelling human decision-making with sufficient fidelity, using conventional software approaches. Current implementations of computer-generated forces within simulations, such as CAEN or ModSAF[4] have proven to be very useful, but do not model human reasoning and also cannot easily model team behaviour. Early applications of the AOP in simulation to represent operational military reasoning, have proved highly effective. This success comes from the capability of agents to represent individual reasoning.

Within Defence, the contemporary trend towards the integration of multirole forces, together with the high cost of live exercises, has required the development of more realistic training environment. However, these synthetic environment have not been able to model the behaviour of the human being involved, other than in a very simple manner. In particular, they find difficulty in modelling team behaviour.

The AOP allow the computer-generated forces in training systems to behave in a more human-like manner, with a much richer set of behaviours, including team responses and dynamic role re-allocation. The result is a more effective training environment with realistic tactical behaviour represented, whilst avoiding the expense of having humans-in-the-loop (HIL) involved to provide this.

Complementing the use of human instructors and teammates with intelligent agents that can take their place when they are not available, could be useful. The intelligent agents cohabit the virtual world with human beings and collaborate (or compete) with them on training scenarios. Intelligent agents have already proven valuable in this role as fighter pilots in large battlefield simulations[3]. The AOP enables the programmer to choose the level of granularity in the simulation of a group by allowing the collapsing of single entities within the team instance.

4.2 Constraints on Network Reliability & Bandwidth

Mostly in the battlefield, the military forces use wireless communication. The wireless commnunication has the drawback of bandwidth, which needs to be resolved. The AOP is a paradigm that enables the programs to move from one host to another, do the processing locally, and return results asynchronously. Thus, it can overcome the barriers posed by network congestion and unreliability.

Soldiers in the battlefield may receive reports from organic sensors and may generate reports based on their own observations. It is important to pass on the critical information gathered by deployed soldiers to other squad members and to echelons above, so that based on the critical information action could be taken as quickly as possible. Agents can be used to disseminate high-priority reports to upper echelons and to other soldiers in a unit based on their information needs.

To minimise load over the very low bandwidth network, an analysis agent determines which soldier in the squad needs a given piece of information (primarily based on location constraints) and agents deliver the reports to the recipients. The delivery agents handle connection failures by retrying at intervals, then informing the sender of a severed connection upon unrecoverable failure. Agents can fulfil an important requirement on robust information dissemination across unreliable networks.

In object-oriented systems, aggregation is defined as a part of relationship in which objects representing components of other objects are associated as an assembly. Aggregation can be classified into two parts–static and dynamic. Static aggregation of objects can be achieved through, for example, inheritance. Static aggregation in object-oriented programs is formed at the compiling phase and irregularities in inheritance or object-containment are detected during this phase. On the other hand, dynamic aggregation refers to enhancing the properties of an object at runtime in unforeseen ways. During an object's execution phase, it can form relationship with other objects of unrelated classes to enhance its functionality. Looking from a mobile agent's perspective, dynamic aggregation helps in reducing the amount of code that goes along with the agent by allowing the agent to attach extra code on need basis, hence, reducing the network bandwidth requirements and also speeding up the process of packing the agent to transfer it from one host to another.

Network class-loading is a feature by which the place server can receive an agent even if it does not have the class information of the agent for de-serialisation. This is an important feature for a mobile agent platform as it relieves the burden of pre-installing the agent and other related class information on all the sites the agent is going to visit. Sometimes, it may not be possible to determine the itinerary of the agent before hand.

The code for the network-class loaders is installed as part of the place server because the place server is responsible for de-serialising an agent. If the place does not find the agent class information for de-serialisation, it invokes the network-class loader to get the class information from the place server where the agent was created. The network-class loader downloads the agent's class-information and defines the class. In the process of defining a class from the class information, the network-class loader may come across some more classes whose information is not available in the local class path. In such cases, the network class-loader recursively applies.

Most applications involving communications over a network use traditional client-server paradigm in which a connection is established between the clients and the server or the data-grams are sent across the network. This traditional approach becomes expensive and unreliable when lots of messages have to be sent between the client and the server, i.e., when the application consumes a lot of network bandwidth. In such situations, sending the client to the server's machine for performing the job locally rather than shouting the commands across the network will be more efficient and reliable. This forms the basis for mobile agents. Mobile agents are software agents that have the basic capability to move themselves from host-to-host and continue execution from the point they stopped on the previous host.

Mobile agents overcome most of the inherent limitations in client-server paradigm. First and foremost, the mobile agent paradigm shatters the very notion of client and server. With mobile agents, the flow of control actually moves across the network, instead of using the request/response architecture of client-server paradigm. In effect, every node is a server in the agent network and the agent (program) moves to the location where it may find the services it needs to run at each point in its execution. For example, the same agent interacts with the user via a GUI to obtain request keys, and then travels to a database server to make its request.

The problem of robust networks is greatly reduced for several reasons. The hold time for connections is reduced to only the time required to move the agent in or out of the machine. Because the agent carries its own credentials, the connection is simply a conduit, not tied to user authentication or spooling. No request flows across the connection, the agent itself moves only once, in effect, carrying a greater payload for each traversal. This allows for efficiency and optimisation at several levels.

4.3 Heterogeneous Data Resources & Environment

Since an agent can travel from a machine of one type to the machine of another type, it is necessary that agent should be platform-independent. This will relieve the programmer of the problems arising due to heterogeneous data sources and environment. Most of the agent architectures are achieving this using JAVA technology, since this technology provides one of the most important features, i.e., platform-independence.

Recent studies have shown a need for compliance between agents developed on various platforms and in heterogeneous environment. For this purpose, a large consortium of research organisations and companies have developed a specification known as mobile agent facility (MAF). Currently, the MAF specification is at its infancy, and still, a lot of refinements are needed to make it a better and complete industrial[6]. To enable the agents of different designers to interact with each other, it is necessary to standardise the basic services that are provided by agent management system. One such standard is FIPA[7-8].

4.4 Information Push, Information Pull, & Sentinel Information Monitoring

When the agents automatically send information to other agents or entities that may need it, it is information push. When agents retrieve relevant information from distributed sources, it is information pull, and sentinel information monitoring, means one or more agents persistently checking for an event or existence of a condition and reacting to its occurrence. These three behaviours are needed, either individually or in combination, in nearly all the military applications[9]. The AOP significantly reduces information dissemination and retrieval latencies. It allow a user to make abstract queries, information requests for which the user could specify query parameters in high-level concepts rather than in exact database schemata. The abstract query mechanism provides mobile agents with tasks to be executed at individual databases. With this capability, operators can avoid exhaustive searches of all data sources, and instead, search the sources that are actually relevant to the query.

5. CONCLUSION

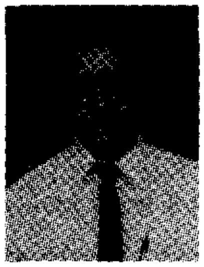
The proposed model of decision support system demonstrates the potential for extensive use of the AOP in Defence domain. Many hurdles like low bandwidth, network reliability, complex autonomous information processing involving large heterogeneous data sources, and heterogeneous operating environment, which are faced during use of traditional OOP and client-server approach, can be overcome using the AOP. The ability of AOP to exhibit social behaviour, reactivity, and pro-activity are additional advantages.

The proposed model of decision support system, which can be introduced phasewise, without affecting existing applications and the environment used in the Defence domain, will certainly be helpful in increasing the coordination between the Defence Forces and reducing the strategic communication gap.

REFERENCES

1. Wagner, G. Agent-object-relationship modelling from agent theory to agent implementation together with EMCRS 2000. *In* Proceedings of Second International Symposium, April 2000.
2. Shoham, Y. Agent-oriented programming. *Artificial Intelligence*, 1993, **60**, 51-92.
3. Wooldridge, M. & Jennings, N.R. Intelligent agents: Theory and practice. *Knowledge Engg. Rev.*, 1995, **10**(12), 115-52
4. http://www.aaii.com.au
5. Jones, R.M.; Laird, J.E. & Nielsen, P. E. Automated intelligent pilots for combat flight simulation. *In* Proceedings of the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98), Menlo Park, CA. AAAI Press, 1998. pp. 1047-054.
6. Johnny, Wong; Guy, Helmer; Venkatraman, Naganathan; Sriniwas, Polavarapu; Vasant, Honavar & Les Miller. SMART mobile agent facility. *J. Syst. Software,* 2001, **56**, 9-22.
7. http://www.agentlab.de
8. http://www.fipa.org
9. Susan, McGrath; Dara, Chacon&Kenneth, Whitebresd. Intelligent mobile agents in the military domain: www.atl.lmco.com

Contributor



Mr Sunil Doiphode received his BE (Computer Science) from the B. N. College of Engineering in 1992. He worked as Lecturer in the B.N. College of Engineering from 1992 to 2000. He joined the DRDO at the Research & Development Establishment (Engrs), Pune, in 2000. His areas of research include: Agent-oriented programming, artificial intelligence, operating system, and office automation. He is a life member of the Computer Society of India and the Indian Society of Technical Education.