Journal of Applied Information Science
11 (2) 2023, 13-20
http://www.publishingindia.com/jais

Agent and Multi-Agent System through Artificial Intelligence with Distributed Environment

Meena Sachdeva[1*], Himanshu Grover[2] and Davinder Kumar[3]

[1]JIMS Engineering Management Technical Campus, Greater Noida, Uttar Pradesh, India.
Email: meena.sachdeva255@gmail.com
[2]BITS Pilani, Pilani, Rajasthan, India.
[3]Micron Electronics, Hyderabad, Telangana, India. Email: davinder991@gmail.com
*Corresponding Author

Abstract: Agent is an autonomous entity and has the ability to act on behalf of others, can understand the changes in the environment and react according to environment with the help of features like mobility, learning ability etc. An agent can be a person, a machine, a piece of software. Agents are autonomous, flexible and active according to the environment. We can say that Agents are acute and suitable for pro-active and social behavior. In this paper discuss about agents, starts from definitions, types of agent, architecture of agent and mobile agent with distributed environment. For developing latest pattern software application we can use software agents. So agent-based computing is best for software development and new revolution in software.

Keywords: Agents, Artificial intelligence, Distributed environment, Information security, Security agents.

I. Introduction

Artificial Intelligence (AI) and agent systems are very closely related to each other [1]. AI is based on the mechanism of intelligence e.g., the ability to learn, plan but Agents are based on integrating these same mechanism. This difference will be seen to all the problems that are associated with agents.

II. Agents Classification

The main classes of agents are defined as follows:

- Reactive agents
- Goal-based agents
- Utility-based agents
- Interface agents
- Mobile agents
- Information-gathering agents
- Multi-agents
- Collaborative agent systems

Some agents are hybrids, which exhibit properties of more than one of the categories listed above [2]. The eventual aim of most intelligent agent research is to develop smart agents, which would be fully autonomous and able to learn and cooperate with other agents.

A. Reactive Agents

Reactive agent (Reflex agent) is a production system where inputs from the environment are compared with rules to determine which actions to carry out. They work on the basis of defined rules [3]. An example of a reactive agent is the automatic mail filter that many e-mail systems now possess. This mail filter examines each e- mail as it arrives and compares it against a set of rules, or templates, and classifies it accordingly. A common use for such systems is to reject so-called “junk mail” or “spam” [4]. A reactive agent does not tend to perform well when its environment changes or when something happens that it has not been told about. Example, an e-mail–filtering system might have problems when it receives an e-mail that is entirely in Chinese. This type of situation can be handled by written new rules but agents can deal with such type of situations.

B. Goal-Based Agents

These are very complicated as compared to reactive agents. This type of agents following a predetermined set of rules, a goal-based agent acts to try to achieve a goal. This can do with searching Technology and through planning. For example, be given the goal of finding pages on the Internet that are of interest to an Artificial Intelligence researcher. This can do with the searching technique and through planning.

C. Utility-Based Agents

These worked as goal-based agents, but they used to increase their utility value with achieving goal. This value can be used as the happiness of the agent or how successful it is being. It may also take into account how much work the agent needs to do to achieve its goals. Let us go to our example from the previous section of an agent that searches for pages on the Internet that are of interest to Artificial Intelligence researchers [5]. The utility-based agent can use knowledge about the Internet to follow the most worthwhile paths from one page to another. This can be done by using heuristic-based search techniques to minimize the time.

D. Interface Agents

An interface agent can be thought of as a personal assistant. These are independent agents; carry out tasks on behalf of a human user. Interface agents collaborate with the user, but do not need to collaborate with other agents although in some cases, interface agents can learn by seeking advice from other agent [6]. For example for using new software package used for help as manual guide. Such an agent has the ability to observe what the user does and make suggestions for better ways to perform those tasks [7]. It is also able to assist the user in carrying out complex tasks, possibly learning as it does so.

E. Mobile Agents

These types of agents are able of moving from one place to another. In the case of mobile robots, this literally means moving in physical space. In the case of mobile software agents, this mobility usually refers to the Internet or other network [8]. An agent that is not mobile is static. Mobile agents travel from one computer to another, gathering information and performing actions as needed on the basis of that information [9]. A computer virus can be thought of as a form of mobile agent, although most viruses are not intelligent, merely autonomous. Without intervention of human these types of agents can work or they work independently [10]. They follow a fixed set of rules that tells them how to infect a computer and how to reproduce. The main advantages of mobile agents are in efficiency [11].

Using a large amount of bandwidth, which can be avoided if the agent is able to physically move to the remote server and query it locally.

F. Information-Gathering Agents

The Information agents known as the information-gathering agents, are usually used on the Internet so these are also sometimes called the Internet agents [12]. An information agent is used to help a user find, filter, and classify information from the vast array of the sources available on the Internet. Information agents may be static or mobile. Some information agents are capable of learning, whereas the behavior of others is fixed. Additionally, information agents can be collaborative or can work independently of other agents [13]. The distinctive feature of an information agent is the function that it provides, rather than the way it works.

G. Multi-Agents

These systems are composed of multi-agents that interact with each other to achieve a common goal. These systems worked on artificial intelligence, economics and sociology.

H. Collaborative Agent Systems

These types of agent systems are Multi-agent systems in which the agents collaborate with each other to achieve goals. This property, of cooperating to achieve a common goal, is known as benevolence [14]. Collaborative agents typically do not have the ability to learn, although some have simple learning abilities [15]. As with multi-agent systems, the idea is that a combination of many simple agents can solve a problem that each agent individually would not be able to solve.

III. Agent Architecture

Agent architecture is the basic study of independent components that carries effective behavior. These range from purely reactive or behavioral architectures that operate in a simple stimulus – response fashion, such as those based on the subsumption architecture, at one extreme, (BDI) model, at the other extreme to more deliberative architectures. In between the two lie hybrid combinations of both, or layered architectures, which attempt to involve both reaction and deliberation in an effort to adopt the best of each approach [16]. So agent architectures can be divided into four main groups: logic based, reactive, BDI and layered architectures [17]. Logic-based (symbolic) architectures draw their foundation from traditional knowledge-based systems techniques in which an environment is symbolically represented and manipulated using reasoning mechanisms. The advantage of this approach is that human knowledge is symbolic so encoding is easier, and they can be constructed to be computationally complete, which makes it easier for humans to understand the logic. The disadvantages are that it is difficult to translate the real world into an accurate, adequate symbolic description, and that symbolic representation and manipulation can take considerable time to execute with results are often available too late to be useful.

A. Reactive Architectures

Implement decision-making as a direct mapping of situation to action and are based on a stimulus – response mechanism triggered by sensor data. Unlike logic-based architectures, they do not have any central symbolic model and therefore do not utilize any complex symbolic reasoning [18]. The key ideas realized this architecture are that an intelligent behavior can be generated without explicit representations and abstract reasoning provided by symbolic artificial intelligence techniques and that intelligence is an emergent property of certain complex systems. Subsumption-designed agents perceive conditions and act, but do not plan. The advantage of this approach is that it will perform better in dynamic environments as well as that they are often simpler in design than logic-based agents. Fig. 1 shows subsumtion architecture for robot navigation.

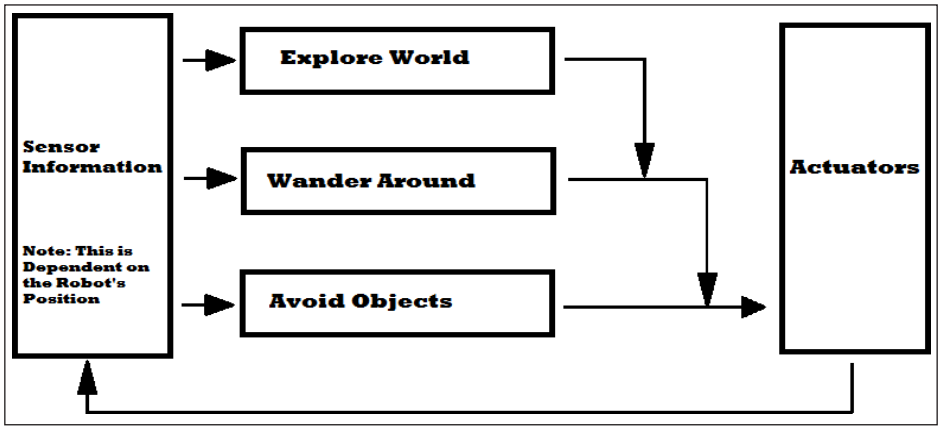

Fig. 1: Subsumption Architecture for Robot Navigation

B. BDI (Belief, Desire, Intention) Architectures

BDI **a**re probably the most popular agent architectures [19]. They have their roots in philosophy and offer a logical theory which defines the mental attitudes of belief, desire and intention using a modal logic. One of the most well-known BDI architectures is the Procedural Reasoning System (PRS). Fig. 2 defines the PRS agent architecture. This architecture is based on four key data structures: beliefs, desires, intentions and plans, and an interpreter.

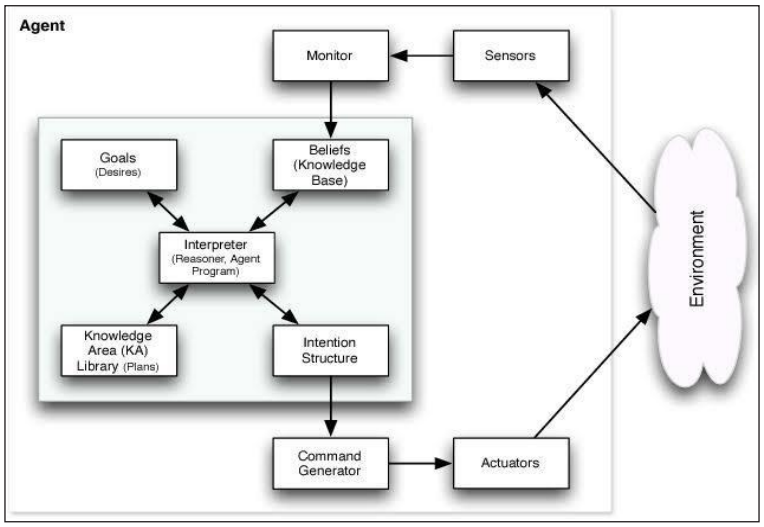

Fig. 2: The PRS Agent Architecture

In the PRS system, beliefs represent the information an agent has about its environment, which may be incomplete or incorrect. Desires represent the tasks allocated to the agent and so correspond to the objectives, or goals, it should accomplish. Intentions represent desires that the agent has committed to achieving [20]. Finally, plans specify some courses of action that may be followed by an agent in order to achieve its intentions. These four data structures are managed by the agent interpreter which is responsible for updating beliefs from observations made of the environment, generating new desires (tasks) on the basis of new beliefs, and selecting from the set of currently active desires some subset to act as intentions [21]. Finally, the interpreter must select an action to perform on the basis of the agent's current intentions and procedural knowledge.

C. Layered (Hybrid) Architectures

Allow both reactive and deliberative agent behavior. To enable this flexibility, subsystems arranged as the layers of a hierarchy are utilized to accommodate both types of agent behavior. There are two types of control flows within a layered architecture: horizontal and vertical layering. In horizontal layering, the layers are directly connected to the sensory input and action output which essentially has each layer acting like an agent. The main advantage of this is the simplicity of design since if the agent needs n different types of behaviors, then the architecture only requires n layers. However, since each layer is in effect an agent, their actions could be inconsistent prompting the need for a mediator function to control the actions.

The vertical layer architecture eliminates some of these issues as the sensory input and action output are each dealt with by atmost one layer each (creating no inconsistent action suggestions). The vertical layered architecture can be subdivided into one-pass and two-pass control architectures [22]. In one-pass architectures, control flow from the initial layer that gets data from sensors to the final layer that generates action output. In two pass architectures, data flow up the sequence of layers and control then flow back down. The main advantage of vertical layered architecture is the interaction between layers is reduced significantly to m 2(n-1). The main disadvantage is that the architecture depends on all layers and is not fault tolerant, so if one layer fails, the entire system fails.

IV. Mobile Agents

Can be also defined as computational software those are capable of roaming different WANs. While on tour, a mobile agent interacts with foreign hosts and gathers information on behalf of its owner who has initiated it. After performing its duties set by the owner, the mobile agent comes back to the originating source/owner [23]. In mobile computing environment, users can access information independent of their location. But accessing this information should not restrict mobility of users. From data management point of view, mobile users can handle only fraction of data since mobile devices are having limited resources. The development of low cost and yet portable mobile devices have enabled mobile users to work from anywhere, at anytime. Distributed Environment itself is a term which covers all types of environments where distribution is done over any of the functional or non functional parts are distributed or over the whole system [24]. It may be distributed systems, client/server system, distributed database management system, distributed computing, remote execution etc. In my work the term “Distributed Environment” is used as I have stressed on the distributed database system which is non-homogeneous (i.e. heterogeneous). There are different significant benefits of using mobile agent in applications those are generally works in distributed environment. It is not so that the applications which are built using the mobile agent concept cannot build without mobile agent (using static agent). But the use of static agent will make the cost on higher side.

A. Mobile Code Paradigms

There are four basic types of Mobile Code Paradigms.

- Client/Server
- Code on Demand
- Remote Evaluation
- Mobile Agents

In case of all the systems mentioned above, the basic elements are:

Data: This basically contains the result-set fetched by the system while in execution.

Code: Code is the section with which different modules necessary for the functionality of the system.

Program Stack: It contains the current status of the program.

(i) Client/Server Paradigm

It is the most widely used paradigm. Here, in this paradigm the services are offered by a server and the service is consumed by one or more, generally remote clients.

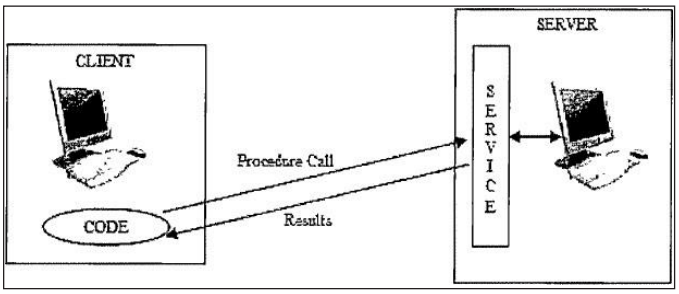

Fig. 3: Client/Server Paradigm

The Fig. 3 depicts the client/server paradigm. Here in the figure, in the client side, the “CODE” form which a procedure in the server is called and the server executes the procedure and returns the results back to the client. Few examples are RPC, Web-Services, CORBA, EJBs etc. [25]. Here, in client/server system, the data element is mobile and both the code and the program stack element are static. The systems, which follow the Mobile Agent and distributed environment [9, 11]. Client/server paradigm, are easy to implement and hence this paradigm is very popular and wide-spreading.

(ii) Code on Demand Paradigm

In this paradigm, the server has different procedures for service to the client and client will get the service from the server. But the service, the client wants, is transferred to the client from the server. When the code for the particular processing is received

by the client, the code is executed to perform the task for the client. Fig. 4 represents the Code on Demand paradigm. Here, in the figure, the "Code-1" is responsible for the request, from the client, for "Code-2" in the server. The server, on receiving the request, transfers the "Code-2" to the client. Client, on receiving the "Code-2", it executes the code to carry out the task. Fig. 4 depiticts the code on demand paradigm. In this paradigm, the data element and the program stack is static but the code element is mobile. Centralized code base and simplicity in updating the software are main advantages of such type of systems.

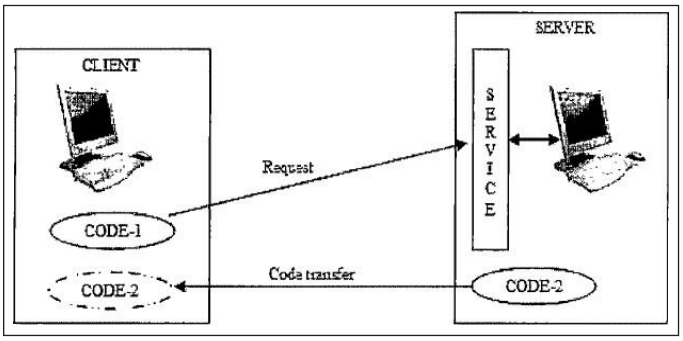

Fig. 4: Code on Demand Paradigm

(iii) Remote Evaluation Paradigm

In this paradigm, the client has the code for execution but transfers the code to the server and the server executes the code and returns the results to the client.

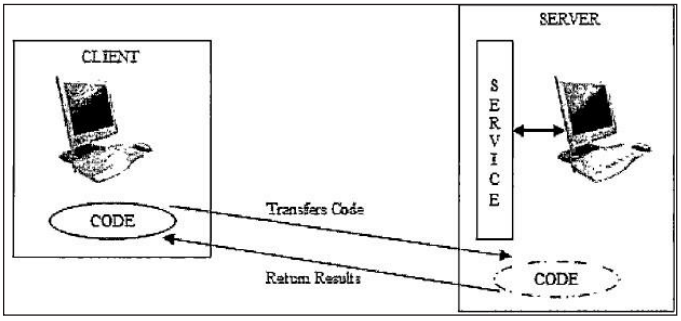

Fig. 5: Remote Evaluation Paradigm

The Fig. 5 represents the Remote Evaluation paradigm'. Here in this paradigm, the "Code" (in the client side), that is to be executed, is transferred to the server and the server, on receiving the "Code" executes it and sends the results back to the client. In this paradigm, the data element and the program stack is static but the code element is mobile.

(iv) Mobile Agent Paradigm

The Mobile Agent paradigm actually derives from two different disciplines and they are Artificial Intelligence and Distributed Systems. The artificial intelligence helped in creation the concept of an agent and the distributed system defines the concept of code mobility across a network. According to standard definitions, mobile agents have everything that a non-mobile agent has; agents are autonomous, reactive, proactive and social. In addition to these mobile agents are also movable, i.e. they can migrate between platforms in order to accomplish the assigned task. In this paradigm, the case lies in remote execution of component. The component sends itself or another on-behalf of itself, to a remote host. The component moves with its code and data. In case of state of the state it may remain intact or modified depending upon the implementation. Unlike the case with remote execution, the component (i.e. mobile agent) will be deciding for itself that whether it wishes to move to an alternate location or not. There are number of advantages of mobile agents over their static counterpart. Thus mobile agents include some of the benefits like reduction in communication costs, limited local resources, easier coordination, asynchronous computing etc. Mobile agents, sometimes called mobile objects, constitutes of Code (behavior), Data, Execution State and Itinerary. These elements are bundled together and are able to move as single unit. In case of stationary objects, which only consist of Code (behavior) and Data. In case of both, the behavior is represented by interfaces but in case of stationary objects as they do not move, the Code and Data can be platform dependent. Mobile agents, on the other hand, can move and therefore, their code, data, execution state and itinerary must all be portable, or at least convertible from and into portable forms.

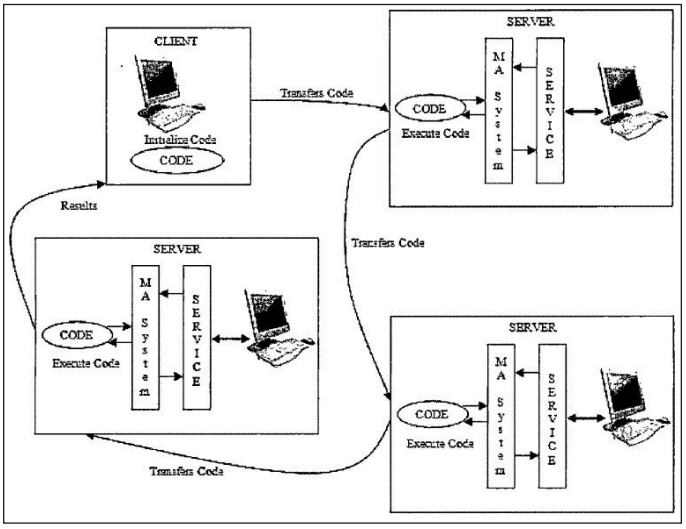

Fig. 6: Mobile Agent Paradigm

Fig. 6 represents the concept of mobile agent paradigm. In the figure, the code in the client in initialized and after initialization the code is transferred to a server. When the mobile code receives the server the Mobile Agent System executes the code. After executing the code, the server then sends the code to another server for execution. And thus after reaching the last server the code is again executed and the overall result of all the executions is sent back to the client.

V. Distributed Environment

A distributed database is logically a single database but actually it spreads in different computers in a network. In case of distributed database there must be multiple DBMSs running at each computer and there exist co-operation between them.

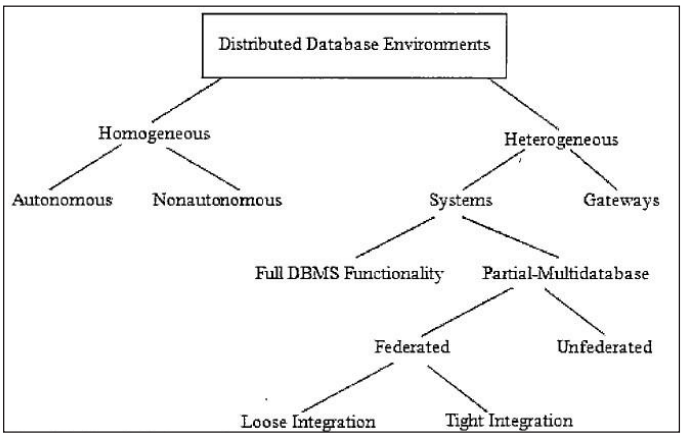

Fig. 7: Distributed Environment

The Fig. 7 represents the range of distributed database environments.

A. Homogeneous Distributed Database

This environment architecture is very simple and therefore it is much easy to design and manage. In this environment all sites/node must have the same DBMS. Also all the sites should agree to cooperate in processing user requests. This type can be divided into two categories:

- *Autonomous:* In autonomous kind, each DBMS at each node/site works independently. They pass messages back and forth to share data updates.
- *Non Autonomous:* In non autonomous kind, there must be a central or master DBMS, who coordinates database access and update across the nodes/sites. But in general, each site/node has to surrender some part of its autonomy in case of right change the schema or these software.

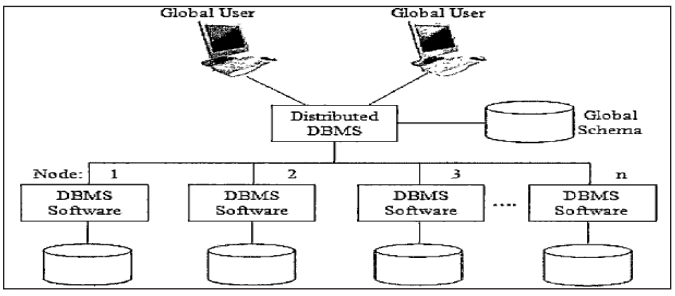

Fig. 8: Non-Autonomous Homogeneous Distributed Database Environment

The Fig. 8 depicts a non-autonomous homogeneous distributed database environment. There are few characteristics those define this kind of environment and they are as follows:

- Data can be distributed across all the nodes.
- At each node/site same DBMS is used.
- The distributed DBMS manages all data.

All users access the database through one global schema (database definition). This global schema is just the union of all the local database schemes.

B. Heterogeneous Distributed Database

In this kind of distributed database environment, the sites/nodes are not bound to use the same schema and software. Since there are differences in schemas at the sites/nodes, therefore problem arises in the processing of queries and transactions. Also, the sites may not be aware of each other. Heterogeneous systems are useful when each of the sites/nodes may have their own hardware, software and data structure which may not be compatible. In heterogeneous system, translations are required to allow communication between different sites/nodes. Systems Supports some or all of the functionality of one logical database. Full DBMS Functionality supports all of the functionality of a distributed database; Partial-Multi database supports some features of a distributed database. Federated Supports local databases for unique data requests. Loose Integration Many schemas exist, for each local database, and each local DBMS must communicate with all local schemas. Tight Integration One global schema exists that defines all the data across all local databases. Unfed rated requires all access to go through a central coordinating module.

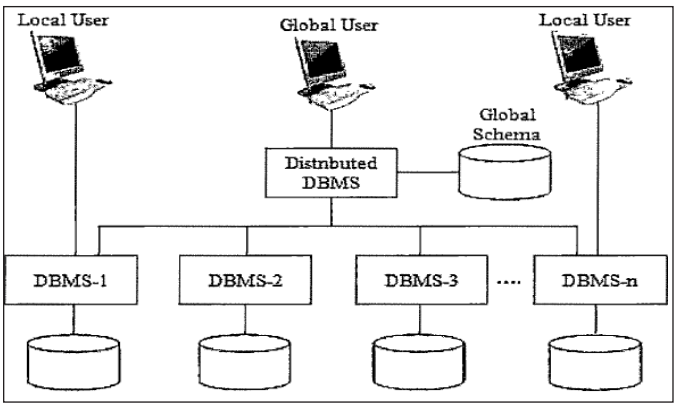

Fig. 9: Non-Autonomous Heterogeneous Distributed Database Environment

The Fig. 9 represents a heterogeneous environment which can be defined by the following characteristics:

- Gateways: Gateways are simple paths created to other databases.

- Data are distributed across all the nodes. Different DBMSs may be used at each node. Some users require only local access to databases, which can be accomplished by using only the local DBMS and schema. A global schema exists, which allows local users to access remote data [9]. The adoption of a particular Distributed Database Environment for the design development of a system is basically depends on the conditions/flexibilities demanded by the system to be developed.

VI. Conclusion

So we conclude that the agents focused on the part of many sub-fields of computer science and artificial intelligence. Agents are being used in a more and more wide variety of applications, ranging from comparatively small systems such as email Filters to large, open, complex, mission critical systems such as air traffic control.

References

[1] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, and A. Patel, "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 9, pp. 2105-2127, 2013.

[2] A.-M. Zou, K. D. Kumar, and Z.-G. Hou, "Distributed consensus control for multi-agent systems using terminal sliding mode and Chebyshev neural networks," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 3, pp. 334-357, 2013.

[3] M. H. Bowling, "Convergence and no-regret in multiagent learning" in *NIPS*, 2004, pp. 209-216.

[4] A. Zidan, M. Khairalla, A. M. Abdrabou, T. Khalifa, K. Shaban, A. Abdrabou, R. El Shatshat, and A. M. Gaouda, "Fault detection, isolation, and service restoration in distribution systems: State-of-the-art and future trends," *IEEE Transactions on Smart Grid*, 2016.

[5] P. Balaji, and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in Multi-Agent Systems and Applications-1.* Springer, 2010, pp. 1-27.

[6] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence.* Prentice-Hall.

[7] L. C. Jain, and D. Srinivasan, *Innovations in Multi-Agent Systems and Applications-1.* Springer, 2010. Egnlewood Cliffs, vol. 25, p. 27, 1995.

[8] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of self-organization mechanisms in multiagent systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 441-461, 2017.

[9] A. P. Garcia, J. Oliver, and D. Gosch, "An intelligent agent-based distributed architecture for smart-grid integrated network management," in *2010 IEEE 35th Conference on Local Computer Networks (LCN),* IEEE, 2010, pp. 1013-1018.

[10] R. Arjun, A. Kuanr, and K. R. Suprabha, "Developing banking intelligence in emerging markets: Systematic review and agenda," *International Journal of Information Management Data Insights*, vol. 1, no. 2, 2021, Art. no. 100026.

[11] H. Rezaee, and F. Abdollahi, "Average consensus over high-order multiagent systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 3047-3052, 2015.

[12] L. Ma, H. Min, S. Wang, Y. Liu, and S. Liao, "An overview of research in distributed attitude coordination control," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 121-133, 2015.

[13] J. Qi, R. Vazquez, and M. Krstic, "Multi-agent deployment in 3-D viapde control," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 891-906, 2015.

[14] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra and its Applications*, vol. 197, pp. 143-176, 1994.

[15] C. Godsil and G. F. Royle, *Algebraic Graph Theory.* Springer Science & Business Media, 2013, vol. 207.

[16] H. F. Ahmad, "Multi-agent systems: overview of a new paradigm for distributed systems," in *2002 Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering*, IEEE, 2002, pp. 101-107.

[17] Q. Liu, L. Gao, and P. Lou, "Resource management based on multi-agent technology for cloud manufacturing," in *International Conference on Electronics, Communications and Control (ICECC)*, 2011.

[18] F. M. Al-Shrouf, "Facilitator agent design pattern of procurement business systems," in *2008 32nd Annual IEEE International Computer Software and Applications (COMPSAC'08)*, IEEE, 2008, pp. 505-510.

[19] J. Fu, and J. Wang, "Adaptive coordinated tracking of multi-agent systems with quantized information," *Systems & Control Letters*, vol. 74, pp. 115-125, 2014.

[20] M. Y. Lee, K. G. Atkins, Y. K. Kim, and S. H. Park, "Competitive analyses between regional malls and big-box retailers: A correspondence analysis for segmentation and positioning," *Journal of Shopping Center Research*, vol. 13, no. 1, pp. 81-98, 2006. [Online]. Available: https://jrdelisle.com/JSCR/2005Articles/JSCRV13_1A4CompetitiveAnalyses.pdf

[21] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE*

Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3416-3452, 2018.

[22] D. Angeli, and P. A. Bliman, "Extension of a result by moreau on stability of leaderless multi-agent systems," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 759-764.

[23] Y. Zhao, G. Wen, Z. Duan, X. Xu, and G. Chen, "A new observer type consensus protocol for linear multi-agent dynamical systems," *Asian Journal of Control,* vol. 15, no. 2, pp. 571-582, 2013.

[24] D. Zhang, L. Meng, X. Wang, and L. Ou, "Linear quadratic regulator control of multi-agent systems," *Optimal Control Applications and Methods*, vol. 36, no. 1, pp. 45-59, 2015.

[25] Z. Li, W. Ren, X. Liu, and M. Fu, "Consensus of multi-agent systems with general linear and lipschitz nonlinear dynamics using distributed adaptive protocols," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1786-1791, 2013.