A Review of Software Agents: An Empirical Study

Sangeeta Arora[1], Dr. P. Sasikala[2]

[1]*K.I.E.T, Ghaziabad, India*

[2]*M.C.N.U.J.C, Bhopal, India*

Abstract: **The agents have taken birth from artificial intelligence that agent use sensors to perceive their environment through actuators. Agents performs task interactively, autonomously and flexibility in complex environments. Now a day, agents are used in industrial and as well commercial applications. In this paper, we have discussed basics of agents and their architecture, development methodologies, framework and implementation platforms. The three agent architectures are discussed i.e. deliberative, reactive and hybrid architectures. The difference between deliberative architecture and reactive architecture has been presented. Where, deliberative architectures are not well suited for dynamic environments. Some important methodologies i.e. GAIA, MASE, PASSI etc. are discussed. Implementation platforms middleware, reasoning and societal platforms are described for agents.**

I. INTRODUCTION

In agent technology, it has been observed gradually a lot of research done on agent theories, architecture, and measurement. This is too considerable to improve in advancement of far above the ground excellence and complex industrial systems. Agents can be used in infinite number of applications in various domains i.e. security system, web agents, in medical applications etc. Even though agents are admired in research area, still not well accepted in industry as the leading development technology, reason behind this is lack of methodologies.

Software Agents have been figure out in Artificial Intelligence for many years. Russel & Norvig [1] stated that an agent use sensors to perceive their environment through sensors, which are take action upon the environment through actuators.

Gilbert [2] et al. characterized the agents in aspects i.e. mobility, intelligence and agency: The intensity of autonomy gives power to agents to converse with agents and entities of the organization; Intelligence is seen as the level of reasoning and gain knowledge of behavior; The level of Mobility is agent's ability to travel throughout the network. S.Franklin and A. Graesser [3] discussed that Agency fall into two categories: Strong agency where behavior is intelligent like human brain; Weak agency in which some form of intelligence exists, use set of pre-programmed rules. In software engineering weak agency agents are used called software agents with the following characteristics (Fig. 1):

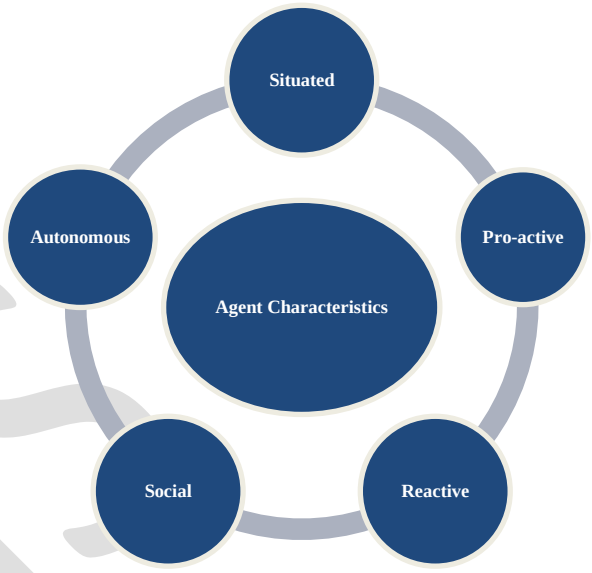

Fig. 1: Characteristics of an agent

a. *Situatedness:* The agents embedded in environment for sensing activities and ready to perform action calculated upon transformation of state.
b. *Autonomy:* Autonomy illustrates that agent act upon different states with no involvement of humans or agents. Agents use their own knowledge for realization towards goal and independently handle its tasks. Agents work independently with the available resources and cooperation with others.
c. *Reactivity:* The response is given by perceiving the environment at the time change occurs in environment. Agent has ability to deduce environment, vital requirements to adapt new knowledge in terms of behavior as time changes.
d. *Pro-activeness:* Pro-activeness shows the goal directed behavior or intelligent reactions as environment changes. Agent reacts intelligently towards goal and learning behavior through tasks to acquire knowledge.
e. *Social Ability:* To achieve the goal, sometimes agent interacts with each other to help. Social ability approached in different forms: cooperation to help each other; competition for initiation and negotiations to for decision making.

II. AGENT ARCHITECTURES

Agent architectures yields package of components i.e. formal specification or design model. Agent Architecture helps to agree on kind of components used and the interactions among them. Maes [4] , Kaelbling [5] , Jennings and Wooldridge [6] stated that architecture show abstract view of system, find out action based on current state and find the new state of an agent. Kim [7] and Rosario Girardi [8] also reviewed existing agent architectures and for the improvement agent behavior is also analysed.

A. Deliberative Architecture

Deliberative architectures have a plan to implement the task with view of environment. In this agent stores the knowledge depends on the method chosen; due to this reason deliberative agents are not good for dynamic environment. Belief-Desire-Intention (BDI) defined by Bratman [9] is an example of deliberative architecture; beliefs as prior knowledge, desire as stimulus or probable preferences and intentions are obligations towards beliefs and desire.

B. Reactive Architecture

Reactive Architectures are best suited for dynamic environment in which decisions are based on environment existing state. It does not possess beliefs and logical formulae to produce new knowledge. The advantage of reactive architecture is that it can work on critical events quickly. This architecture does not create best plan for dynamic environments due to lack of prior knowledge. Subsumption architecture defined by Brooks [10] is the example of reactive architecture. A finite state machines with different layers which are further connected to sensors to perceive the environment to take action.

C. Hybrid Architecture

Hybrid Architecture has the best of both architectures i.e. deliberative and reactive architecture. It uses deliberative architecture for making plans and use previous knowledge; reactive architecture for interactions in dynamic environment. This type of architecture work in hierarchical manner; upper layer work as deliberative for long term goal directed behavior and planning; lower layer as reactive architecture to obtain information from the environment.

The difference between deliberative and reactive architecture is shown in Table 1.

III. FRAMEWORK FOR AGENTS

A framework is proposed for agents to requirements engineering because of intended behavior i.e. i* framework Yu E. [11]. Agents are able to achieve goal on the basis of beliefs or their current state due to their strategic relationships. Agents depend upon other agent to achieve the goal; it also shows weakness of agent as well opportunity. Agents are part of an organization to calculate their strategic position in a societal environment. When multiple autonomous component with deliberative concern; it might be emphasizing interdependence on another components. The i* framework represented with two forms: Strategic Dependency and Strategic Rationale.

Table 1: Difference between Deliberative and Reactive architectures

Deliberative	**Reactive**
Learn from prior experience	Do not learn from past experience
Prepare a plan towards goal-directed behavior	Do not have plan
Have beliefs and formulae for plan	Do not have beliefs and formulae
Not suited for dynamic environment	Best suited for dynamic environment
Based on Reasoning Process	Do not learn from past experience
Build decision from existing knowledge	Build decision by perception of environment

Strategic Dependency shows the dependency among two actors; where dependent actor is known as depender and other is called dependee. Both actors have dependency to achieve the goal using their knowledge. Strategic Dependency is of four types: goal, task, resource and softgoal. In Goal dependency, depender and dependee rely on each other for the attainment of goal. In task dependency, depender has the plan to execute the task but all activities cannot be carried out without help of dependee. In resource dependency, depender acquire the resources of dependee only but decisions are not dependent on dependee. In softgoal dependency, depender shares some tasks to achieve softgoal where softgoal is like a subgoal to achieve a goal.

IV. METHODOLOGIES FOR DEVELOPMENT

For the expansion of system with agents, we need to follow the process which follows all the phases. All required aspects of software development system is illustrated in the software engineering methodologies. A modelling language is used to confine all the requirements, interactions, goal achievements of product. Comparing with OO methodologies, AO methodologies are building up by researchers as well as industry driven. AO methodologies are inclined to human organization because of agent can also participated in different role and work together with other agents. Due to this different human organization concepts are accepted in agent based system like role of agent, social dependency of agents, and position of agent in organization.

An Agent Oriented methodology pivots around the agent's role, interaction, autonomous activity and respond to environment. These methodologies help in the development of agent based software but no control on implementation language. Some of methodologies are discussed below:

A. GAIA

Zambonelli [12] et al. proposed first methodology for development of agents, good for dynamic environments.

GAIA spins around with phrase dependability, action and set of rules within the organizations. Dependability describes the role of agent which further depends upon responsibilities assigned to agents. Actions are the tasks performed by agent and set of rules define how agents are interact with each other. Like other methodologies GAIA begin through the analysis phase, to gather the role and tasks of an agent. This is further continued with design dependability, actions of agent. Detailed design specifications of agent based systems are not done technology specific.

B. Prometheus

Padgham and Winikoff [13] proposed methodology for agent based systems called Prometheus, which incorporate the different phases for development of system. In system specification phase, first goals and tasks are identified. In architectural design phase, specification of agents and interactions are given. In the detailed design phase, planning is done for data structure, tasks and events for agents of the system. The UML's scenarios and sequence diagrams are used in Prometheus for identifying interactions. UML and AUML is used to define the dynamic behavior of the system. The roles identified are used for plot the tasks and coupling in the system specification phase.

C. Tropos

Bresciani et al.[14] proposed methodology emphasize on the requirement analysis at the early stage. i* framework is used by Tropos methodology to suggest agents and their roles, social dependency of agent on each other which includes goals further divided into softgoal, task and resources. Requirement Analysis is base phase of all software development phase helps to implement the system. Initially Tropos work with programming framework only do not hold implementation phase. Later Morandini et al.[15] extend four phases used in waterfall or spiral model for development i.e. sequential and iterative. The four phases are requirements for early and late, detailed and architectural design. A dependency graph is designed using strategic dependency model to show dependency among agents. A strategic rational graph is designed using strategic rational modal to show the goals, resources and tasks. Detailed design helps to design the social behavior i.e. communication among agents.

D. MASE

Multiagent System Engineering (MaSE) approach is introduced by Deloach[16] et al. having two fundamental stages analysis and design. In the analysis phase, goals are captured using goal hierarchy, sequence diagrams for interactions and goals are refined. In the design stage, agent class diagram, conversation diagram, agent architecture diagrams, deployment diagrams are devised. DeLoach and Wood [17] also designed the agentTool system for supporting and implement the MaSE methodology. This tool automatically draws from the agent architecture from analysis phase where roles and tasks are described. Automatic verification is done for conversation is also provided by this tool.

E .PASSI

Massimo Cossentino et al., [18] proposed a methodology for agent societies specification and implementation (PASSI) which is used to design and expand agent based societies from requirement to code. It integrates design model as well object oriented software engineering and intelligence come within reach of UML syntax. The PASSI methodology covers viewpoint of human aspects of agent based system also in the different steps like requirement gathering, social relationship, architecture, code and implementation. PASSI methodology incorporates the following model:

- System Requirement Model: for the requirements with viewpoint of agent
- Agent Society Model: shows the communication and dependence among agents
- Agent Implementation Model: architecture of an agent in view of classes and functionality
- Code Model: for the implementation of agent based systems
- Deployment Model: install the system

V. TOOLS, PLATFORMS AND PROGRAMMING LANGUAGES (FRAMEWORKS)

Different tools are required for building up agent based systems to implement the agents and their architectures. Several tools are introduced step-by-step designing agent applications for successful implementation of the system. These tools are classified into two classes: one for the development of applications is agent platform and second for execution of system. The development phases of agent based system inclined towards agent specific methodologies using the specific types of models and hold precise to agent tools. In the object oriented tools, two broad categories are classified: one is UML Computer Aided Software Engineering tools (CASE) and other one is integrated development environment (IDE) e.g. NetBeans. CASE tools are used for designing help towards the implementation and IDE tools helps to coding phase. With the help of IDE testing and deployment also can be done. All tools work towards agent specific characteristics or platform.

Various fundamental services suggested for hosting agent on standardized communication by an agent platform. and depict readymade communication mechanism. The intelligent agent architecture is promoted towards industry for cooperation, development for agent based applications by Foundation for Intelligent Physical Agents (FIPA). Different associated groups i.e. firms or educational institutes participated all the way through. The implementation of an agent is characterized internal architecture where as co-operation among agent platform represent social architecture of agent.

A large amount of agent platforms are existing in the market i.e. industrial, open-source now days. We are discussing summarized view of agent platforms which are categorise on the basis of classification of them. There are three main categories of agent platforms which are middleware, reasoning and societal view (Fig. 2).

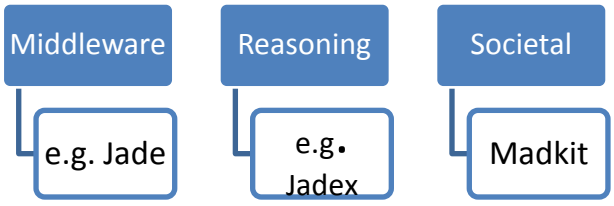

Fig. 2: Agent Platforms

Middleware

A layer linking an application with operating system to offer generic services which recycle with diverse categories of applications called middleware by Coulouris et al. [19]. Different features of agents i.e. interoperability, scalability, robustness and mobility. Where interoperability is one of important feature, agents cooperate with each other to achieve the goal. Interoperability is recognized by the adherence of Foundation for Intelligent Physical Agents (FIPA) standards. Middleware are not specific to programming languages rather focused on model for task.

The Telecom Italia Lab (TILAB) developed middleware for agent paradigm is Java Agent Development Environment (JADE) in 1998 Bellifemine et al.[20] . JADE shared out as free software in February 2000 by TILAB. Motorola and Telecom Italia form the governing board for JADE to fulfil the purpose to support the development and acceptance of JADE as middleware. JADE is developed in Java language.

JADE agents follow the behavior aligned architecture where behavior concerns to a task and provide encapsulation for particular role. The data stores are shared among behavior for communication, which shows all processing results of communication done among them. A unique thread is created for every agent which is used for non-preemptive scheduling. FIPA patterns accumulated in the JADE platform and container to host the agent with two exclusive agents i.e. Agent Management System (AMS) and Directory Facilitator (DF). DF agent keeps the details of all available agents in the platform. AMS agent is created self to manage the platform and responsible to create other agents, destroy agents and containers and stop the platform. FIPA Agent Communication Language (ACL) is used for message among agents.

JADE is enriched with variety of tools which are accessible for developing different agent applications, debugging and administration. Integrated development environment is used same as java i.e. eclipse, netbeans without limitations. Remote Monitoring Agent (RMA) monitors the life cycle of an agent; activate or deactivate the message transfer protocols on containers. Dummy agents can also be used for stepwise execution and monitoring the agent behavior. Sniffer is also available to visualize the communication among agents like UML sequence diagram.

Reasoning Platform

Reasoning platform is lined around the inner reasoning method of agent and makes the arrangements to endow with prospect for the efficient execution of agent behavior. Reasoning platforms are characterized to believe in psychological theories for describing realistic human behavior. The application is developed for performing tasks like humans rather abstract theories. This is the reason that agents require new programming languages which extend, refine and interpret the basic theory. Intended position is preserved for implementation of agents because of philosophy used by McCarthy [21]. Jadex is the example of reasoning platform.

The development of Jadex framework is done at the university of Hamburg since 2003 Pokahr et al. [22]. Jadex is an extension to JADE for the implementation of reactive and deliberative agent architectures; representation of mental states in JADE subsequent the Belief-Desire-Intention (BDI) model. Jadex isolates reasoning and execution infrastructure for agents. Jadex has been make use of recognize applications in diverse fields e.g. scheduling, simulation etc.

As discussed above, Jadex works with BDI model; fundamental notion is belief, goal and plan. Belief is the knowledge of an agent which is used to react on the perception of the environment. Goals are put in for the agent's feedback and deliberation method; behavior is determined by beliefs, goals and plans. The outlook of an agent is like black boxes, which sends and receive messages. Goals, received messages and internal actions provide as key for the effect of agent and deliberation. The process of deliberation transmits events of new plan or existing plans. The plans already running may us, alter the beliefs and pass messages among agents, generate goals and cause internal events. In Jadex XML is used for the arrangement of the agent composition according to permission tags and attributes of an agent. Other points like plans are programmed in Java accessible through a framework API and communication language is FIPA compatible using JADE.

Social Platform

Social agent platform is revolving around the coordination and cooperation part of agent based systems. Social platform is not concerned with individual agent behavior; target is aligned on the group behavior of agents. Multi-agent development kit (MadKit) framework is an example of social platform for integrated behavior.

The MadKit is open source platform constructs by Gutknecht et al. [23]. In MadKit framework is sticked to the Agent/ Group/ Role (AGR) model and treat organization as structure.

MadKit is micro-kernal based framework which consist of withhold facilities for agent life and group organization and transferring messages. MadKit centred towards organizational aspect of agents rather precise agent architecture of and agent. An agent is an autonomous in MadKit to communicate messages and occupy agents in groups. MadKit supports different languages Python, Java and Jess for implementation. Madkit provides the different tools i.e. source code editor, designer tool and group observer tool to show organizational structure, groups and agents available, also allows the visualized conversation among agents.

VI. CONCLUSION

An agent technology is concerned about growth to amplify interest in perspective of agent. Agents can be used in multiple complex areas and accepted in industrial applications. Many facts are discussed in this paper i.e. agent architectures, methodologies, framework and tools are discussed. Agent architecture is represented in three forms: deliberative, reactive and hybrid architectures. Deliberative architecture is based on BDI model so it learns from experience whereas reactive architecture is best suited for dynamic environment and do not learn from experience. Hybrid architecture uses features of both architectures. Some important methodologies are discussed, in which some methodology follows all the phases of development but some are focused only analysis and design. Three agent platforms are focused which are middleware, reasoning and societal. Middleware provides the generic services to applications; reasoning platforms are based on psychological theories like human behavior; societal platforms work on social aspects of agents.

REFERENCES

[1]. Stuart Russell, Peter Norvig, "Artificial Intelligence: A Modern Approach", Prentice Hall, Second Edition, 2003

[2]. Gilbert D, Aparicio M, Atkinson B, Brady S, Ciccarino J, Grosof B, O'Connor P, Osisek D, Pritko S, Spagna R, Wilson L, 1995, IBM Intelligence Strategy, IBM Corporation

[3]. S. Franklin, A. Graesser, Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96), volume 1193, Berlin, Germany, 1996. Springer-Verlag.

[4]. P.Maes, "The Agent Network Architecture (ANA)", in "SIGART Bulletin", Vol. 2, Issue No.4, 1991. pp.115–120.

[5]. L. P. Kaelbling, "An Architecture for Intelligent Reactive Systems", in "Reasoning About Actions and Plans – Proceedings of the 1986 Workshop", M.P. Georgeff and A.L. Lansky Eds., Morgan Kaufmann, 1986. pp. 395–410

[6]. M. Wooldridge, N. R. Jennings, "Intelligent Agents: Theory and Practice", in "The Knowledge Engineering Review", Vol. 10 Issue No. 2, 1995. pp. 115–152.

[7]. Kim On CHIN, Kim Soon GAN, Rayner ALFRED, Patricia ANTHONY & Dickson LUKOSE, "Agent Architecture : An Overview", Transactions on Science and Technology, 2014, Vol. 1, pp 18-35

[8]. Rosaria Girardi, Adriana Leite, "A survey on Software Agent Architecture", IEEE Intelligent Informatics Bulletin, 2013, Vol 14, pp. 8-20

[9]. Bratman, M. 1987. Intention, Plans, and Practical Reason. Harvard University Press

[10]. Brooks, B. 1989. How To Build Complete Creatures Rather Than Isolated Cognitive Simulators. In Architectures for Intelligence, 225–239

[11]. Yu, E. 1995. Modelling Strategic Relationships for Process Reengineering, PhD, University of Toronto, Department of Computer Science.

[12]. Zambonelli, F., Jennings, N., and Wooldridge M. 2003. Developing Multiagent Systems: the Gaia Methodology. ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3

[13]. Padgham, L., and Winikoff, M. 2004. Developing Intelligent Agent Systems: A Practical Guide. John Wiley and Sons.

[14]. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A. 2004. Tropos: An Agent-Oriented software development Methodology, Autononuous Agents and Multi-Agent Systems, vol 18, 203-236

[15]. Morandini, M., Nguyen, D.C., Perini, A., and Susi, A. 2008. Tool-Supported Development with Tropos: The Conference Management System Case Study. In the proceedings of 8th International Workshop on AGENT ORIENTED SOFTWARE ENGINEERING (AOSE 07) Revised Selected Papers. LNCS 4951 Springer

[16]. Scott A. DeLoach, Analysis and Design using MaSE and agentTool, 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001) Miami University, Oxford, Ohio, March 31 – April 1, 2001

[17]. Scott A. DeLoach & Mark Wood. Developing Multiagent Systems with agentTool, Intelligent Agents VII - Proceedings of the 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL'2000). Springer Lecture Notes in AI, Springer Verlag, Berlin, 2001

[18]. Massimo Cossentino, Colin Potts, A CASE tool supported methodology for the design of multi-agent systems, The 2002 international conference on Software Engineering Research and practice SERp'02(2002)

[19]. Coulouris, G.F., Dollimore, J., and Kindberg, T. 2005. Distributed Systems. Addison-Wesley

[20]. Fabio Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing Multi-Agent Systems With JADE", John Wiley & Sons Ltd,2007

[21]. McCarthy, J. 1979. Ascribing mental qualities to machines. In: M. Ringle (Editor): Philosophical Perspectives in Artificial Intelligence. Humanities Press, 161–195

[22]. Pokahr, A., Braubach, L., and Lamersdorf, W. 2005. Jadex: A BDI Reasoning Engine. In: R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni: Multi-Agent Programming: Languages, Platforms and Applications, Springer, 149–174

[23]. Gutknecht, O., Ferber, J., and Michel, F. 2001. Integrating Tools and Infrastructures for Generic Multi-Agent Systems. In Proceedings of the Fifth International Conference on Autonomous Agents, 441-448.