Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 1 Issue 4, November - 2014

Agent-Oriented Software Engineering Characteristics and Paradigm

Ardavan Ashabi
Advanced Informatics School
Universiti Teknologi Malaysia (UTM)
Kuala Lumpur, Malaysia
ardavan.ashabi@gmail.com

Khalil Salah
Advanced Informatics School
Universiti Teknologi Malaysia (UTM)
Kuala Lumpur, Malaysia
salah.kh@gmail.com

***Abstract*—Nowadays, agents as well as multi agent systems have been of research interests among computer science communities. This is particular in the capturing of structure in the natural way and in the behavior of complex systems. Software project problems are interpreted by actions evolving upon the paradigm in software engineering. This paper upon a brief introduction regarding key aspects of agent and agent based computing, explains the reason for these new direction in computer engineering field. Agent orientation has become a necessity in software engineering. The paper also discusses crucial engineering agent characteristics and the challenges to be faced.**

Keywords—Agent-oriented; Agent-based Computing; Software Engineering Paradigm.

I. INTRODUCTION

An agent is defined as an encapsulated system of computer which is located in an environment which is able to have flexible and autonomous action in there for meeting its design objectives [1].

In practical applications, environment is open. It's dynamic and complex. A consequence is that association among the elements can't be fully foreseen at the time of compile. Also the inherent organizational structure of the system should be clearly represented.

Proper abstractions, methodologies and tools are needed for correctly engineer such kind of applications. Agents are considered as the model for engineering distributed systems for future which are open and complex:

Open: Elements can join or leave the environment that is dynamic. The way the operating conditions get altered is unpredictable.

Complex: The software has huge number of elements which interact following the complicated interaction protocols; all agents have a partial view of the environment. Also there is no control that is centralized.

Agent Based Computing is a synthesis of Computer Science (CS) and Artificial Intelligence (AI) [2]. Agent's notion usually appears in various contexts of computer science, usually with various meaning. In the context of AI or Distributed AI, agents as well as multi agent systems are usually exploited as a way for tackling complicated problems and for developing software systems that are intelligent [3][4][5]. Main acceptation is AI and Distributed AI. Here agents are normally exploited as a way for developing special purpose systems showing some sort of intelligent behavior. Specifically we take into consideration of agents and relevant concepts as general purpose abstractions which are useful for the programming of software systems as a whole, conceptually extending programming that is object oriented with aspects which are effective for tackling certain major challenges in modern day software development [6].

Agent oriented paradigm inherits object oriented one. AI paved way for the evolution of the agent concept. Agents' nature is being social, reactive and being proactive. Agent oriented software is developed using agent oriented technologies [7].

Agent technology has got much attention in the past few years as a result of which the industry has been showing interests for developing its own products. Despite various developed agent theories, architectural systems and languages as well as successful applications that are agent based, only slight works for specifying ways for developing applications using agent technology is being done. Role played by such technologies is in assisting every stages of life cycle of an agent based application which includes management too [8].

Complicated nature of the process of software development has resulted in developing strong natural abstraction with which for modeling and developing complicated systems. Procedural abstraction, types of abstract data and objects are such abstraction's examples [9]. In the last 20 years, due to the complicated nature of software projects, agent concepts which originated from AI has been considered for devising a new paradigm to handle complicated systems [1],[10],[11], [12],[13],[14],[15].

There are two main viewpoints for agents. The first one is powerful AI viewpoint where the agent is considered proactive and intelligent. There should be P2P discussed rather than doing client to server computing. The next viewpoint is the less powerful software engineering element which has threads of

execution that are internal. This can be engaged in complicated and state interaction protocols [16].

II. Reasons that Agent-Based System Is Seen as a Critical New Direction

There are various reasons for the present intensity of interest, but for sure one of the crucial one is the agent's concept as an autonomous system which has the ability for interacting with different agents for satisfying the system's design objectives, is a natural one for designers of software products. Similar to the way that we can understand various systems are being composed of passive objects that are essential, so we can also understand various others which are being made of agents that are interacting and semi-autonomous.

A sure question to ask is why the agents and multi agent systems are considered crucial new direction in this engineering field of software:

A. Natural Metaphor

Similar to the way that various domains can be conceived of having many interacting objects that are passive essentially, various others can be conceived as interacting and purposeful agents who are active too.

B. Distribution of Data

In most software systems, it is impossible for identifying a locus of control; rather an overall control is the system is passed across various computing nodes that are frequently distributed geographically. For making such systems work properly these nodes got to be capable of autonomous interaction among them- they must agents.

C. Legacy Systems

A natural method for incorporation of legacy systems into modern distributed information system is to agentify them: for wrapping them with no agent layer, which shall enable them for interacting with different agents other than this.

D. Open Systems

Most systems are open. That is, it is difficult for knowing during the time of design exactly which elements comprise the system and how these elements will be used for interacting among one another. For operating effectively in such systems, the ability for engaging in autonomous decision making which is flexible is very important [17].

III. Agent-Oriented Software Engineering

A. The requirement for Agent-Oriented Software Engineering

1) Software engineering is crucial in discipline such as software systems and processes. Reliability of the approaches is on the abstraction sets and on relevant technologies as well as tools.

2) Agent based computing which introduces abstractions that are novel, which needs clarification of the group of essential abstractions. It also needs new tools production and methodology adaptations.

3) Novel and specific software engineering methods that are agent oriented are required [16].

B. Key Software Engineering Agents' Characteristics

Agent based solutions can't be applied in every situations. One critical factor for successful agent oriented software engineering is the thus identification of the application needs which shows an agent based solution. What are the major characteristics which shows an agent based way is appropriate [18]?

Software agent can be defined as a computer systems located in an environment which acts on its user's behalf and it is characterized by various properties [19].

Many researchers agree that autonomy is an agent's crucial property. It's precisely the autonomy of agents that define the agents [20]. Also the cooperation between various software agents can be useful for attaining their targets [19]. As per [21] the most usual way in using the term agent is for denoting hardware or software based system of computers which enjoys the properties such as: social ability, being pro-active, reactivity and autonomy. In [22] major concepts in the definition was identified which was adapted from [21]. They are autonomy, social ability, reactivity and pro-activeness. In [23] an intelligent system defines as the one that has autonomy and enjoys having it, having social ability, pro-activeness and reactivity. He also emphasized that the components put forward by other researchers such as veracity, rationality, learning, mobility and benevolence also need to get good focus [ibid].

Agents possess various characteristics in different combinations. Table 1 enumerates and gives definition for all characteristics adopted for this research's purpose [24].

TABLE I. Characteristics of Software Engineering Agents

Characteristics	Definition
Autonomy	It means that the agent can act without direct intervention by humans or other agents and that it has control over its own actions and internal state (Sycara, 1998).
Reactivity or situatedness or sensing and acting	It means that the agent receives some form of sensory input from its environment, and it performs some action that changes its environment in some way (Chira, 2003; Sycara, 1998).
Proactiveness or goal directed behavior	It means that the agent does not simply act in response to its environment; it is able to exhibit goal-directed behavior by taking the initiative (Chira, 2003; Wooldridge & Jennings, 1995; Odell, 2000).
Social ability	It means that the agent interacts and this interaction is marked by friendliness or pleasant social relations; that is, the agent is affable, companionable or friendly (Odell, 2000).
Coordination	It means that the agent is able to perform some activity in a shared environment with other agents (Odell, 2000). Activities are often coordinated via plans, workflows, or some other process management mechanism (Odell, 2000).

Characteristics	Definition
Cooperation or collaboration	It means that the agent is able to coordinate with other agents to achieve a common purpose; non-antagonistic agents that succeed or fail together (Odell, 2000).
Flexibility	It means that the system is responsive (the agents should perceive their environment and respond in a timely fashion to changes that occur in it), pro-active and social (Jennings *et al.,* 1998).
Learning or adaptivity	It means that an agent is capable of i) reacting flexibly to changes in its environment; ii) taking goal-directed initiative, when appropriate; and iii) learning from its own experience, its environment, and interactions with others (Chira, 2003; Sycara, 1998).
Mobility	It means that the agent is able to transport itself from one machine to another and across different system architectures and platforms (Etzioni & Weld, 1995).
Temporal continuity	It means that the agent is a continuously running process, not a "one-shot" computation that maps a single input to a single output, then terminates (Etzioni & Weld, 1995).
Personality or character	An agent has a well-defined, believable "personality" and emotional state (Etzioni & Weld, 1995).
Reusability	Processes or subsequent instances can require keeping instances of the class 'agent' for an information handover or to check and to analyze them according to their results (Horn *et al.*, 1999).
Resource limitation	An agent can only act as long as it has resources at its disposal (Horn *et al.*, 1999). These resources are changed by its acting and possibly also by delegating (Horn *et al.*, 1999).
Veracity	It is the assumption that an agent will not knowingly communicate false information (Wooldridge & Jennings, 1995; Wooldridge 1998).
Benevolence	It is the assumption that agents do not have conflicting goals and that every agent will therefore always try to do what is asked of it (Wooldridge & Jennings, 1995; Wooldridge 1998).
Rationality	It is the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit (Wooldridge & Jennings, 1995; Wooldridge 1998).
Inferential capability	An agent can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility; goes beyond the information given, and may have explicit models of self, user, situation, and/or other agents (Bradshow, 1997).
"Knowledge-level" communication ability	The ability to communicate with persons and other agents with language more resembling humanlike "speech acts" than typical symbol-level program-to-program protocols (Bradshow, 1997).
Prediction ability	An agent is predictive if its model of how the world works is sufficiently accurate to allow it to correctly predict how it can achieve the task (Goodwin, 1993)

Source: adopted from Georgakarakou, C. E., & Economides, A. A, 2009, p.5

Concluding we can express that the traits of software engineering agents can be divided into 2 major groups: the first one is the fundamental traits such as being pro-active and autonomy, interactivity and situatedness. The second one is traits that are additional such as locality, mobility, adaptation, learning and openness [16], [ibid].

C. Agent-Oriented Software Engineering (AOSE) Paradigm

Paradigm in software engineering evolved action for interpreting issues of a software project. Various paradigms are used these days. Selecting paradigm is based in features regarding the project, application types, devices and type of controls needed. Paradigm distinguishes in abstraction and method of approach. It is tough for building best quality software. In software engineering, various kinds of programming paradigms are developed. Binary language was used at first for writing programs.

For making programming simpler, assembly languages got developed which used mnemonic codes. Procedural languages are as per the unit and scope concepts. Program writing in procedural languages is much easy. Every successive development either makes the engineering procedure easier as it claims or extends the complicated nature of applications which be built feasibly. A new paradigm evolved from the advancing turn before. For example, to object oriented from procedural and to component oriented from object oriented. Similarly agent oriented paradigm evolves [7].

Agent orientation emerges as a new paradigm for the construction of software systems. Newer systems types are getting developed as per the concept of software agent. As per a definition [17], software agents are located where they sense environment, they are autonomous and have control upon their own actions and states of internal and can act with no straight intervention from people and are flexible, that is responsive to environmental changes, oriented towards a goal, opportunistic and take steps initially and are also social, that is they interact with other agents that are artificial as well as human beings for completing their targets and helping others.

Agent orientation provides a new way of thinking regarding software and is its construction. Compared with the previous software paradigm that was dominant, called as object orientation, agent orientation gives a top level abstraction for thinking regarding characteristics and behaviors of software systems. It can be considered as a part of the current trends regarding in-depth interactivity in conceptions of software programming and construction of software systems [25], [26], [27], [28].

When developing string and robust software capabilities is evident basically to the enterprise of software and IS, it is also evident that we require effective methods for determination of need and requirement for specific application setups so that the correct system can be developed foe meeting the needs. A software systems' success depends on the needs and how good they are resolved during development stage [28].

The needs engineering community is active in developing newer concepts and methodologies [28], [29], [30]. This study shows that agent orientation can be as an important paradigm that shifts for needs engineering as for the construction of software. Models and languages are important for needs engineering. They enable the proper type of knowledge for expression for supporting the correct type of reasoning and analysis. As the requirements and contexts of needs engineering vary, advances in modeling and languages are also required for responding to the changes. Conventional modeling methods reflect the mechanistic view of the world in which they concentrate on specifications of behavior which are detailed, known and completely controllable. Current systems are environments are more complex and dynamic, accommodate much domestic freedom and initiative and got to cope-up with limited control as well as limited knowledge [28].

A proper concept of agent got to be developed for serving as a central construct in a latest type of modeling and analysis for responding to current day needs. Most like the concepts of object and concepts of activity which plays vital role in initial days' modeling paradigms, the concept of agent can be instrumental for putting forward a shift towards an even richer ontology that is socially oriented which is required for characterizing and analyzing current days systems and environments [ibid]. Agent oriented systems are string, even flexible and also rebuts that the traditional software systems [ibid].

Agent Oriented Software Engineering (AOSE) paradigm shows an interesting way to analyze, design and build complicated software systems and it quite suits the new development of software needs [15].

Yoav Shoham proposed the paradigm of AOSE in the year 1990 [13]. This was based on a societal computation view [13], [14]. Artificial Intelligence is the major source for this paradigm [1], [31]. Distributed Artificial Intelligence is precisely the source for this [32], [33], [15].

In AOSE agents are more of computer science and software engineering than of AI [1]. Agent oriented paradigm got multiplied many times in the last 20 years and even though it was initially limited to academic level researches, it got the industry's interests in the last few years [31], [33], [15]. It has to be noted that almost after 10 years of its introduction, this paradigm's progress faced severe transformation, which certain researchers refers as the gateway to new generation methods in software engineering [33], [34], [15]. Hence this paper on AOSE paradigm strengths indicates the essentiality of its use [15].

IV. Discussion And Conclusion

Agents as well as multi agent systems are at present one of the most discussed topics in research area among computer science communities. The natural method of capturing the structure as well as behavior of complicated systems has stimulated this interests among computer science researchers particularly. This resulted in making agent oriented software engineering, which paved way for a new paradigm in software engineering [35].

Agent based computing is a promising method for application development in complicated domains. Anyhow even after much research being done in this stream, various challenges got to be resolved still such as making agent based comp ting's acceptance wide in the practice of software engineering, for turning agent oriented software abstractions in to tools that are practical to face the complicated nature of the current day areas of application [36].

Today's complicated systems and our expectations regarding the have rose to a level where a social paradigm for modeling as well as analysis is of appropriate than the conventional paradigms [28].

A needs modeling method focusing on a concept of agent can give string new methods for characterizing and analyzing the links and interactions between the various semi-autonomous system entities and in the environment. As per literatures regarding the varying requirements of needs modeling, we put forward the crucial characteristics for software engineering agent which are desirable for a concept of agent methodology for software engineering- reactivity of termed as situatedness, autonomy, acting or sensing, goal oriented behavior of being pro-active, collaborating or cooperation, coordination, adaptability or learning, flexibility, social ability, temporal continuity, rationality, limitation of resource, reusability, veracity, inferential capability, benevolence, communication ability and ability of prediction [ibid], [24].

References

[1] M. Wooldridge, Agent-Based software engineering, IEE Proc. Software Engineering 144 (1) (1997) 26-37.

[2] Alvaro Magri, AOSE - Agent Oriented Software Engineering, ppt.

[3] N. R. Jennings. An agent-based approach for building complex software systems. Commun. ACM, 44(4):35–41,2001.

[4] M. Wooldridge. An Introduction to Multi-Agent Systems. John Wiley & Sons, Ltd, 2002.

[5] S. Russell and P. Norvig. Artificial Intelligence, A Modern Approach (second edition). Prentice Hall, 2010.

[6] Ricci, A., & Santi, A. (2011, September). Agent-oriented computing: Agents as a paradigm for computer programming and software development. In FUTURE COMPUTING 2011, The Third International Conference on Future Computational Technologies and Applications (pp. 42-51).

[7] Arora, S., Sasikala, P., Agrawal, C. P., & Sharma, A. (2012, September). Developmental approaches for Agent Oriented system—A critical review. InSoftware Engineering (CONSEG), 2012 CSI Sixth International Conference on(pp. 1-5). IEEE.

[8] Iglesias, C. A., Garijo, M., & González, J. C. (1999). A survey of agent-oriented methodologies. In Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings (pp. 317-330), Springer Berlin Heidelberg.

[9] Wooldridge M, Jennings NR, Kinny D (1999). A methodology for agentoriented analysis and design, In Proceedings of the Third International Conference on Autonomous Agents (Agents 99), 69–76, Seattle, WA.

[10] Genesereth MR, Ketchpel SP (1994). Software agents, Communications of the ACM, 37, 7, pp. 48-53.

[11] Jennings N, Wooldridge M (1996). Software Agents, IEEE Rev. 17-20.

[12] Jennings NR, Wooldridge M (2000). Agent-Oriented Software Engineering, In Handbook of Agent Technology (ed. Bradshaw J.),AAAI/MIT Press.

[13] Shoham Y (1990). Agent-Oriented Programming, Technical Report STAN-CS-1335-90, Computer Science Department, Stanford University, Stanford, CA 94305.

[14] Shoham Y (1993).Agent-Oriented Programming, Artif. Intell. 60(1):51-92

[15] Akbari, O. Z. (2010). A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance. J. Comput. Engg. Res, 1(2), 14-28.

[16] Onn Shehory, (2008) ,Agent-Oriented Software Engineering, IBM Haifa Research Lab

[17] Jennings, N.R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, 1, 7-38.

[18] Wooldridge, M., & Ciancarini, P. (2001). Agent-oriented software engineering.Handbook of Software Engineering and Knowledge Engineering, 1, 507-522.

[19] Chira, C. (2003). Software Agents, IDIMS Report, 2/21/03, http://pan.nuigalway,ie/code/docs/agents.pdf

[20] Alonso, E. (2002). Al and agents: State of the art. Al Magazine,23(3), 25-29.

[21] Wooldrige, M., & Jennings, N. R. (1995). Inteligent agents: theory and practice. The Knowledge Engineering Review 10(2), 115-152.

[22] Jennings, N.R., Norman T.J., & Faratin, P.(1998). ADAPT: An Agent-Based Approach to Business Process Management ACM SIGMOD Record, 27(4), 32-39

[23] Wooldridge, M. (1998). Agent-based computing. Interoperbale Communicatop Networks, 1 (1), 71-97.

[24] Georgakarakou, C. E., & Economides, A. A. (2009).Software Agent Technology: an Overview Application to Virtual Enterprises

[25] Newell, A. (1982) The Knowledge Level. Artificial Intelligence 18: 87-127.

[26] Bobrow, D.G. (1991) Dimensions of Interaction: AAAI-90 Presidential Address. AI Magazine 12(3): 64-80.

[27] Wegner, P. (1997) Why Interaction Is More Powerful Than Algorithms, Communications of the ACM., 40(5): 80-91. May 1997.

[28] Yu, E. (2001). Agent orientation as a modelling paradigm.Wirtschaftsinformatik, 43(2), 123-132.

[29] van Lamsweerde, A. (2000) Requirements Engineering in the Year 2000: A Research Perspective. Proc. Int. Conf. on Software Engineering, June 2000, Limerick, Ireland.

[30] Nuseibeh, B. A. & Easterbrook, S. M.. (2000) Requirements Engineering: A Roadmap. Proceedings, 22nd International Conference on Software Engineering (ICSE'00), Limerick, Ireland, 5-9 June, 2000. IEEE Computer Society Press.

[31] Debenham JK, Henderson-Sellers B (2002). Full lifecycle methodologies for agent-oriented systems – the extended OPEN process framework, In Proceedings of Agent-Oriented Information Systems (Eds. Giorini P, Lespreance Y, Wagner G, Yu E), Toronto pp. 87-101

[32] Bond AH, Gasser L (1988). A Survey of Distributed Artificial Intelligence, Readings in Distributed Artificial Intelligence, Morgan Kaufmann Publishers: San Mateo, CA.

[33] Henderson-Sellers B, Giorgini P, Bresciani P (2003). Enhancing Agent OPEN with concepts used in the Tropos methodology, in Proceedings of the Fourth International Workshop Engineering Societies in the Agents World, Imperial College London, UK.

[34] Dam KH, Winikoff M (2003). Comparing agent-oriented methodologies, Proceedings of the 5th Int Bi-Conference Workshop on Agent- Oriented Information Systems (AOIS), Melbourne, Australia.

[35] Lind, J. (2001, January). Issues in agent-oriented software engineering. InAgent-Oriented Software Engineering (pp. 45-58). Springer Berlin Heidelberg.

[36] Zambonelli, F., & Omicini, A. (2004). Challenges and research directions in agent-oriented software engineering. Autonomous Agents and Multi-Agent Systems, 9(3), 253-283.