ISSN No. 0976-5697

Volume 4, No. 8, May-June 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Agent Vs Object with an in-depth insight to Multi-Agent Systems

Adnan Ghazi Abuarafah
Faculty of Computer of and IS,
Umm Al- Qura University, Makkah, SA
agabuarafah@uqu.edu.sa

Hussam Aleem Mohammed
Faculty of Computer of and IS,
Umm Al- Qura University, Makkah, SA
mohdhussamaleem@gmail.com

Mohamed Osama Khozium*
Department of Engineering and applied science - computers, MCC-UQU.
Umm Al-Qura University, Makkah, Saudi Arabia
osama@khozium.com

Abstract: In this paper we illustrate the meaning, definitions and importance of intelligent software agent by contrasting agents with objects. Agents are indeed considered to be an evolutionary step forward from objects. The paper is dedicated to visualize the importance and awareness of the agents and multi-agents and will give suitable examples on them.
There are many interesting works showing a methodological approach for Multi-Agent system development. These approaches must be compared to identify the best possible Multi-Agent methodology. Furthermore, this paper highlights the multi agents structure, methodologies and common applications and provide surveys that allow researchers/developers to determine the directions in which agent-oriented methodologies are best suited to achieve goals of a particular project or system. Agents provide software designers and developers with a way of structuring applications around autonomous, communicative components and offer a new and often more appropriate route to the development of complex computational systems, especially in open and dynamic environments.

Keywords: Multi-agent methodologies; Agent environment; Intelligent agent; Prometheus; agent Percepts; agent actions.

I. INTRODUCTION

As we all know very well that software has ever more become part of our daily life and as a result, more complex problems are being faced by the traditional conceptions of software.

In order to overcome these complex problems, the requirements engineering community suggested the need to go beyond such conceptions. Many researches worked on this issue and are pointing to the need to deal with wider aspects in order to be able to understand and model requirements for these systems. [1, 2, 3].

By moving a step ahead from objects towards agents, the above mentioned complex problems can be solved. In order to support this statement, this paper focus mainly on the usage of agents instead of just working with objects and there is a comparison of agent and objects.

A. *Definition of Agent:*

An **agent** is nothing but a computer system which is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. Wooldridge distinguishes between an agent and an intelligent agent, which is further required to be reactive, proactive and social. Agents comprises mental attitudes such as intentions, beliefs and goals. [4]

Agent-based systems offer enhanced functionalities, greater flexibility, and better security, reliability and robustness. The intensifying agent based technology exhibits autonomy and sociality and is proactive. This made the applications shift from the object oriented technology to agent based systems. [5]

Agents are autonomous entities that can interact with their environments. Objects and Agents are distinct enough to treat them differently. When a system is to be designed, it can be chosen as a thought-out mixture of both approaches. In some sense, the burden of getting along with other technologies is a problem for both the agent and object camps. For example, transport policies, directory elements, communication factories, transport references. An agent interacting with an environment is shown in the Figure-1 below.

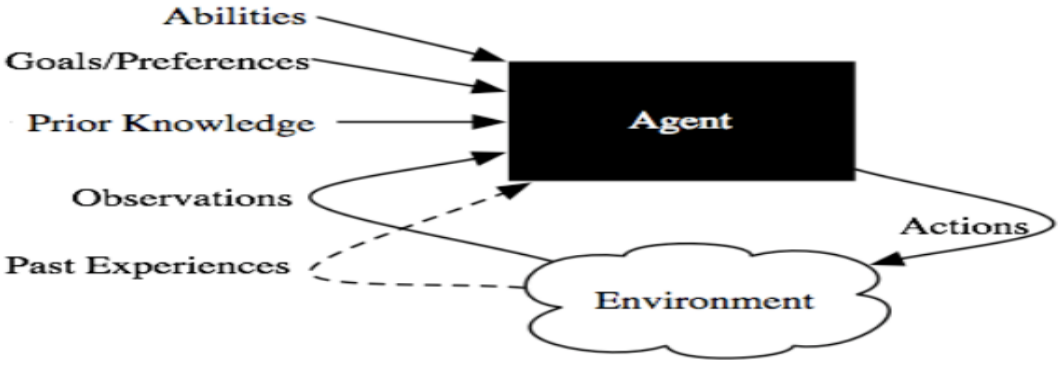

Figure 1 : An agent interacting with an environment

Some Facts about Agent-based Computing

a. An agent is a computer system that is capable of flexible autonomous action in dynamic, unpredictable, typically multi-agent domains.
b. Intelligent agents are helping astronomers detect some of the most dramatic events in the universe, such as massive supernova explosions.
c. Multi-agent systems offer strong models for representing complex and dynamic real-world environments. For example, simulation of economies, societies and biological.
d. Environments are typical application areas.
e. Agents provide software designers and developers with a way of structuring applications around autonomous, communicative components. They offer a new and often more appropriate route to the development of complex computational systems, especially in open and dynamic environments.
f. Working with Rolls Royce, Lost Wax has developed Aerogility, a multi-agent system to help business managers better understand the complexities of the aerospace aftermarket. [6]

B. Definition of Object:

Objects are defined as collections of operations that share a state and determine the messages (calls) to which an object can respond. [7]

"Object-orientation" in simulation refers also to the execution of models, i.e., whether the experimentation with the model happens as message passing between objects. Some object-oriented simulation systems are object-oriented with respect to both dimensions, some, particularly those, that allow also for continuous simulation, support an object-oriented model design but forego an object-oriented execution of the model. [8]

Objects represent passive elements, whereas agents represents active elements in the software system. Agents and objects perform and coordinate their actions in dynamic environments to accomplish the organizations' goals.

Object can also be reactive, and have an implicit goal as that of agents. However, they are not proactive as they cannot have multiple goals and of these goals being explicit and persistent. More of such important advantages and usage of agents over objects will be discussed in this paper with some experiments.

This paper is divided into six sections. The first section is this introduction. The second section discusses the related work. The third section is about the environment types and examples. The fourth section gives the differences between agent vs. object. The fifth section is about the structure and methodologies of the Multi-agent system and its comparison. The last section concludes the work and discusses the future of the research.

II. RELATED WORK

James Odell discusses some of the differences and similarities between agents and objects and lets you decide which viewpoint you want to choose. He mentioned that the agent-based way of thinking brings a useful and important perspective for system development, which is different from—while similar to—the object-oriented way. [10]

However, he brings out only the philosophical differences. Uhrmacher illustrates both the technologies - agents and objects. He gives importance to both the technologies and focused on the usage of both of them [7] However, from this paper, you will get a comparison of agent vs. objects with details on Multi-agent system and will give the reasons of using agent-oriented software design for modeling and simulation that have spawned a plethora of research activities and implementations.

Luiz Marcio et al. gave a practical approach that uses a well defined and complex problem producing specifications using agent/goal orientation and object orientation could guide us to understand better the strengths and weaknesses of each approach. [5]

Silva Viviane, et al 2003 proposed taming agents and objects in software engineering and presented a conceptual framework that provides a conceptual setting for engineering large-scale MASs based on agent and object abstractions. The identified set of abstractions is organized in terms of a unifying framework, providing software engineers with a deeper understanding of the fundamental concepts underpinning agent and object notions and their relationships. Objects are viewed as abstractions to represent passive elements, while agents provide a means of representing active elements in the software system. In addition, a set of additional abstractions is provided to model situations where organizations of cooperating agents and objects perform and coordinate their actions in dynamic environments to accomplish the organizations' goals. [11]

Wooldridge proposed GAIA in which the foundation of analysis is based on a Object-Oriented design method called Fusion, from which it borrows terminology and notations. Their developing process consists of analysis, architecture design and detailed design. Later Tooli and Asaadi proposed GAIA methodology which is used for the analysis and design of agent-based systems. They used an attribute-based evaluation framework which addresses four major areas of an agent-oriented methodology: concepts, modeling language, pragmatics and process. [9] Although GAIA has a clear semantic and understandable modeling language, GAIA doesn't support several features which include traceability and consistency checking There should be some improvement in GAIA like integration of these features into a supporting tool. Although GAIA provides architectural design and requirements analysis, it should provide software quality assurance.

Moreover, there will be detailed review of the structure and methodology of agent-based software technology and Multi-agent systems that will help in getting a clear viewpoint on why agents can be used more widely than compared to objects and finally we will propose the best suitable Multi-agent methodology that can be selected by the Researchers/Developers.

III. ENVIRONMENT TYPES AND EXAMPLES

An intelligent software agent firstly perceives its environment via sensors, then make use of its actuators to accomplish the goals. The examples mentioned in the below table-1 will give a clear idea about why to go a step forward from objects to agents in order to achieve the intricate goals.

A. Examples of Environments & Agents:

Table 1 : Example of Agents [12]

AGENT	Environment	Goals	Percepts	Actions
Intelligent House	• occupants enter and leave house • occupants enter and leave rooms • daily variation in outside light and temperature	• occupants warm, • room lights are on when room is occupied • house energy efficient	signals from • temperature sensor • movement sensor • clock • sound sensor	• room heaters on/off • lights on/off
Automatic Car	• streets • other vehicles • pedestrians • traffic signals / lights / signs	• safe • fast • legal trip	• camera • GPS signals • speedometer • sonar	• steer • accelerate • brake.
Mail Sorting Robot	conveyor belt of letters	route letter into correct bin	array of pixel intensities	route letter into bin
Medical diagnosis system	Patient, hospital	Healthy patient, minimize costs	Symptoms, findings, patient's answers	Questions, tests, treatments
Part-picking robot	Conveyor belt with parts	Place parts in correct bins	Pixels of varying intensity	Pick up parts and sort into bins
Satellite image analysis system	Images from orbiting satellite	Correct categorization	Pixels of varying intensity, color	Print a categorization of scene
Refinery controller	Refinery	Maximize purity, yield, safety	Temperature, pressure readings	Open, close valves; adjust temperature
Interactive English tutor	Set of students	Maximize student's score on test	Typed words	Print exercises, suggestions, corrections

B. Types of environments:

There are several flavors of Environments. [12] The principal distinctions to be made are as follows:

a. Accessible vs. inaccessible:

The environment is said to be accessible if the sensory apparatus of an agent gives it an access to the complete state of environment. The sensors detects all aspects that are relevant to the choices of action. The agent may not keep any internal state to keep track of the world and so *accessible environment* is the convenient one.

b. Deterministic vs. nondeterministic:

The environment is said to be deterministic if the next state of the environment and the actions selected by the agents are completely determined There are no issues of uncertainty in an accessible, deterministic environment for an agent. However, if the environment is inaccessible, then it may appear to be nondeterministic and this is factual if the environment is complex, making it hard to keep track of all the inaccessible aspects. Hence from the agent point of view, it is better to think of an environment as deterministic or nondeterministic.

c. Episodic vs. non-episodic:

An episodic environment is the one in which the experience of the agent is divided into episodes. Each of the divided episode consists of the agent perceiving and then acting. The agent's actions quality depends on only the episode as the subsequent episodes do not depend on what actions occur in previous episodes. The agent does not need to think ahead and so these environments are much simpler.

d. Static vs. Dynamic:

An environment is said to be dynamic for an agent if it can change while an agent is deliberating, otherwise it si said to be static. The environment is said to be semi dynamic if it does not change with the passage of time but the performance score of an agent does.

e. Discrete vs. Continuous:

An environment is said to be discrete there is a limit in the number of percepts and actions that are distinct and clearly defined. Chess can be considered to be discrete as there are fixed number of moves on each turn. Taxi driving can be considered to be continuous as the location and speed of the taxi and other vehicles sweep through a range of continuous values.

So the most challenging environments are inaccessible, nondeterministic, non-episodic, dynamic, and continuous.

IV. AGENT vs. OBJECT

A software Agent comprises two basic properties - autonomous and situated in an environment. The first property of the Agents - being autonomous means that agents are independent and make their own decisions. This is one of the properties that distinguishes agents from objects.

The second property of Agent - being situatedness does not constrain the notion of an agent very much since virtually all software can be considered to be situated in an environment.

The types of environments are the one that make agents differ from object. Unlike objects, agents are used in an environment which is challenging like dynamic, unpredictable and unreliable.

a. ***Dynamic Environments:*** Environments which change rapidly are the *dynamic environments.*
b. ***Unpredictable Environments:*** As the name reflects, these environments are *unpredictable* as it is not possible to predict the future states of the environment. This is because it is not possible for an agent to have perfect and complete information about their environment as the environment is being modified in ways beyond the agent's knowledge and influence.
c. ***Unreliable Environments:*** The environments are unreliable because the actions that can be performed by an agent may fail for reasons which are beyond an agent's control. For example, a robot attempting to lift an item may fail for the item being too heavy or for other reasons.

Objects cannot have multiple goals and so they are not proactive. So the proactiveness is another property that make agent different from objects.

Agents must be robust to recover from the failures due to the challenging environments. In order to achieve this robustness, there should be flexibility. And these two properties of agents make it differ from objects.

From the above comparison of agents and objects, we can conclude that an Intelligent Agent is a software that comprises the following properties

a) Autonomous – independent, not controlled externally
b) Situated – exists in an environment
c) Proactive – persistently pursues goals
d) Reactive – responds (in a timely manner!) to changes in its environment
e) Robust – recovers from failure
f) Social – interacts with other agents
g) Flexible – has multiple ways of achieving goals

In addition to the above mentioned properties, there are other properties of agents which we regard as less central and are important only for certain agent applications.

In order to achieve goals, we may need the agents to be rational. The property of being rational is that an agent should be smart and do not perform 'dumb' things such as simultaneously committing to two courses of action that may result in a conflict. For example, spending money for a holiday and at the same time, spending money on the car. A detailed analysis of what is meant by 'rational' can be found in the work of Bratman. [13] This analysis forms the basis of the Belief-Desire-Intention model for software agents. [14]

One definition of agents (*strong agency*) takes these various properties, and are viewed as having *mental attitudes* such as goals, beliefs and intentions.

V. MULTI-AGENTS

Multi-agent technology is one of the main research fields of distributed artificial intelligence. Agents are used to design and implement complex distributed applications as it have certain properties such as mobility, learning, and autonomy. A network of such cooperating agents each covering a well-defined and restricted part of the solution is a multi-agent system. The task of the multi-agent system is to control and monitor the agents in spite of agents being autonomous. [15]

A. Definition of Multi-agent System:

Multi-agent System (MAS) became a significant research field of distributed artificial intelligence. An agent society of multi-agent is MAS. MAS is also a kind of distributed independent system. Through interactive agents, MAS implement its expressiveness. MAS focus on the research on how to coordinate multi-agent's goal, knowledge, program and strategy with the goal of solving problem or taking action together [16]. MAS has huge market with very wide field of application. [17]

Multi-agent systems offer strong models for representing real-world environments with an appropriate degree of complexity and dynamism. For example, simulation of economies, societies and biological environments are typical application areas. The use of agent systems to simulate real-world domains may provide answers to complex physical or social problems that would be otherwise unobtainable, as in the modeling of the impact of climate change on biological populations, or modeling the impact of public policy options on social or economic behavior. Agent-based simulation spans: social structures and institutions to develop plausible explanations of observed phenomena, to help in the design of organizational structures, and to inform policy or managerial decisions; physical systems, including intelligent buildings, traffic systems and biological populations; and software systems of all types, currently including ecommerce and information agency. [6]

In multi-agent systems, an additional layer of software components is expressed as objects and collections of objects which provides infrastructure that embodies the support for agents composed of object parts

B. Application Research on Multi-agent:

Based on the relevant literatures at home and abroad, several main application fields of multi-agent are listed below. [17]

a. Industry:

One of the activist and the earliest fields of application of Multi-agent is the application in industry. Multi-agent was firstly used in process management.

b. Transportation:

It is suitable to use multi-agent technology in transportation because of the distributed characteristic of the transportation control system.

c. Information Processing:

The demand of information management increased with the increase of bulk information. Information overload came into view due to the Lack of effective tools for information management. The Information filter, Information management, information search and collection was the approach to deal with information overload. Information agent is used in information management to respond to the user requests. [18]

d. Electronic Commerce:

The development of Internet and electronic commerce emerged in the later of 20th century. The Electronic commerce allowed online shopping and selling products by using credit cards.

e. Network:

In case of network management, the system based-on agent contribute to control network and complex system, failure prediction, load balancing, information synthesis and malfunction analysis.

f. Software Development:

Software agent develop multi-agent system by making use of computer Distributed Intelligent Forecasting,

g. Decision-making and Calculating:

The system forecasting can be resolved using the characteristic of multi-agent.

h. Social Simulation:

Multi-agent technology can be looked as experimental tools of sociology.

i. Medicine:

The demands of Medicine application increases with the development of computer science technology increases steadily and these medical applications based on agent is inevitable.

j. Entertainment:

Agent can be utilized into interactive theater, computer game, and other modes of entertainment. [17]

C. Multi-agent methodologies and its comparison/selection:

This section will focus on various methodologies of Multi-agent system. We will then propose comparison/directions to the Researchers/Developers in which agent-oriented methodologies are best suited to achieve goals of a particular project or system.

a. Prometheus:

Prometheus is a methodology which defines a detailed process to specify, design and implement intelligent agents systems. The basic idea behind Prometheus is that it can be easily used by specialists as well as common users. Prometheus is focused on systems that use belief–desire–intention (BDI) agents [23]. The Prometheus is divided into three phases: System Specification, Architectural Design and Detailed Design where System specification identifies the basic system roles through the definition of perceptions, actions and shared data objects. The Architectural Design defines the agents and their interactions in the system. The Detailed design gives detail information about the agent internals.

The Prometheus modeling generated 10 different diagram types. Stakeholders Diagram, Scenarios Diagram, Goal Overview Diagram and Roles Diagram are generated by System Specification. The Data Coupling Diagram, Agent-Role Coupling Diagram, Agent Acquaintance and System Overview Diagram were generated in the Architectural Design. Finally, the Agent Overview Diagrams and Capability Overview Diagrams were generated in the Detailed Design. [23]

b. TROPOS:

Tropos is a software development methodology focus on the requirements aspects allowing a better understanding of the environment involved in the system operation. It uses the framework [24] that describes the actors, goals and dependencies among the actors.

Tropos is divided into 5 major phases namely Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation. Early Requirements allows understanding of a problem by studying an organizational setting and the output is modeled by Strategic Dependency and Rationale Dependency. Late Requirements also generates artifacts, but describing the system-to-be within its operational environment, along with relevant functions and qualities. Architectural Design is where the system's global architecture is defined in terms of subsystems, interconnected through data, control and other dependencies. Architectural design gives the choice of the organizational architectural style, the architecture modeling and the application of social patterns. Detailed Design gives the behavior of each architectural component is defined in further detail and were generated an UML diagram with stereotypes and an AUML sequence diagram representing the interaction between agents. The Implementation phase was not worked in the experimentation. Tropos modeling generated 8 artifacts in the experimentation.

c. MaSE:

MaSE stands for Multi-agent System Engineering proposed by Deloach et al.(2001). MaSE is an object-oriented methodology that supports analysis and design phases using agent-orientated techniques.

MaSE methodology provides developers guidance from requirements to implementation. [19] . MaSE can also be considered a powerful methodology in terms of cooperative agents concepts, definition of autonomy, proactively and autonomy reason and the agent concepts are centered in the Roles Diagram and in Goal orientation. [22]

The development process consists of two main phases: analysis and design where in each step related models are created. A series of steps are provided to model the system in each phase. Models in one step produce outputs that become inputs to the next step, which supports traceability of the models across all of the steps.

a) The analysis phase consists of three steps: capturing goals, applying use cases, and refining roles.

Capturing goals identifies High-level goals from requirements analysis. These goals are then decomposed into sub goals and collected into a tree-like structure. Applying use cases generates use-cases and their corresponding sequence diagram. Refining roles involves role refinement which main task is to map goals into roles where every goal in the system needs a delegated role.

b) The design phase consists of four steps: creating Agent classes, constructing conversations, assembling Agent classes, and system design.

Agent classes creates Agent classes and their interactive behavior and Agent class is recognized. Constructing a conversation helps designers to construct conversation models used by Agent classes. The assembling Agent class step creates Agent class internals. In the System design, Agent classes are instantiated into actual Agents.

d. Masup

MASUP is a Rational Unified Process (RUP) extension [25] that focuses on multi-agent systems development. The

methodology aim is to systematically identify the applicability of an agent solution during the modeling phases. Analysis and Design disciplines are required by MASUP. The agent solution identification occurs on the analysis and design disciplines through a heuristic over activity diagrams. After the identification that an agent solution is appropriate for the problem at hand, MASUP proposes different diagrams to capture agent characteristics. The methodology is fully compatible with RUP and the non-agent part of the system can be modeled using the traditional RUP techniques.

D. Comparative Analysis

All discussed methodologies provide a certain degree of maturity in terms of process definition. Upon careful review, Prometheus and MASUP are considered to be the best processes structure through the precise definition of roles, activities and artifacts. Tropos and MaSE do not exhibit the same firmness for presenting the process structure. For a better use of a software development methodology, we conclude that a process should be clearly defined. [23]

None of the methodology support adaptation, composition, learning or mobility representation. The future agent-oriented methodologies must address an open representation issue in this matter. However, it does not happen with the organization structure modeling. All the methodologies were concerned in a certain level with agent relationships representation. Only two methodologies, MASUP and Tropos were considered interesting in this factor since they provide mechanisms to represent agents' hierarchy.

In terms of models traceability Prometheus seems to be the better methodology. The other three methodologies do not have a specific traceability discipline. However, in software development, traceability is still an open issue. Since multi-agent systems require more aspects to be modeled, traceability techniques should be largely enhanced for those systems. There should be a start of Traceability right in the requirements capture. MASUP and Tropos are most suitable in this matter as these methodologies have a well-defined requirements phase. The other approaches assume that the requirements capture should be done outside their scope.

An agent-oriented approach does not solve all the problems. Prometheus, Tropos and MaSE presume that an agent solution will be applied from the beginning. MASUP has a heuristic to guide role identification throughout the methodology. Any methodology does not cover the aggregation of roles in agents. Guidelines or clear techniques must be established by the new methodologies to identify roles and agents during the modeling phases.

One of the most significant characteristic of multi-agent systems is Communication and interaction. All the discussed methodologies employ special modeling to signify interaction among agents. On the other hand, most methodologies neglect agent internal representation. Agent internals are captured only by Prometheus. There should be proposal of new approaches that shows how the internal agent elements get influenced by the external behavior expected by software agents. Also, the new approach should be able to show which elements are predictable to be found on a software agent.

In terms of implementation platform, methodology should be independent. Implementation independence is maintained by all the discussed methodologies. Prometheus has some studies to translate its representation to the Jack platform. Though the better choice for agent-oriented methodologies is implementation independence, it is interesting the existence of case studies that integrates a methodology to a well-known implementation infrastructure.

Agents should interact in the organization in a way that their common goal can be achieved. Organization process workflow modeling is important to map the agents own goals to the society common goal through a workflow that coordinate agents actions. This factor is outside the scope of all the methodologies examined. This comprises an important improvement opportunity for new agent-oriented methodologies. [23]

There are mechanisms to signify the messages exchanged by agents in all methodologies. The only methodology that uses a proprietary format instead of message representation standards is Prometheus. The designers can reuse previous work results to model the messages in agent communication if standards are used in message representation [23]

VI. CONCLUSION

This paper has an argument on the importance of agent-oriented computing to be a suitable software engineering model when compared with objects for the design, analysis, and development of many modern software systems. This paper provides a realistic review of agent-oriented mechanisms which support requirements of engineering in today's complex application environments. The limitations of the object theories and their abstractions are not powerful enough to examine the new issues in achieving goals. This can be resolved by the companies and researchers which are now investigating how agents can contribute to the mastering of the complexity of modern large-scale systems. Agents can become more widely available and useful if migration strategies are provided sooner. Software developers has composed agents from objects thereby building the infrastructure for agent-based systems which act as an support system used for OO (object oriented) software system. For example, agents can be reasonably expressed as objects. These might include agent names, agent communication handles, agent communication language components (including encodings, ontology, and vocabulary elements), and conversation policies.

We then moved towards Multi-agent technology that has been used in various aspects in the field of society with the development of network technology. The ability of solving complex problem of the system can be improved with the application of multi-agent technology. Moreover, our surveys allow Researchers/Developers to determine the directions in which agent-oriented methodologies are best suited to achieve goals of a particular project or system. To conclude the improvement opportunities from the comparison done in section-5.4, we propose that an agent-oriented methodology should:

a. Use modeling standards;
b. Be clearly defined with good documentation and case studies;

c. Define a traceability process among the artifacts produced;
d. Establish clear techniques or guidelines to identify roles and agents during the modeling phases;
e. Model organization structure with relationship types other than message exchanging and authority;
f. Capture agency characteristics such as adaptation, learning, mobility and composition;
g. Organization workflow definition.
h. Agent internals modeling.

With the developments in the agent-oriented methodologies, research will keep determining the directions in which agent-oriented methodologies are best suited to achieve goals of a particular project or system. Hence, there are various future works that can be done in this area. Moreover for the future work, we may integrate one innovative example/idea into this intelligent agent system.

VII. REFERENCES

[1]. Paolo Bresciani, Fausto Giunchiglia, John Mylopoulos and Perini Anna, 2004. "TROPOS: An Agent-Oriented Software Development Methodology," Journal of Autonomous Agents and Multi-Agent Systems, 8, 203–236.

[2]. Van Lamsweerde, 2001. "Goal-Oriented Requirements Engineering: A Guided Tour," Proc. of 5th IEEE Int. Symp. on Requirements Engineering.

[3]. Annie Antón, 1996. "Goal-Based Requirements Analysis," Second IEEE International Conference on Requirements Engineering (ICRE '96), Colorado Springs, Colorado, pp. 136-144.

[4]. Jennings Nicholas R, Michael J. Wooldridge, 1998. "Applications of intelligent agents". In Agent Technology: Foundations, Applications, and Markets (eds. Jennings NR and Wooldridge MJ), Chapter 1, pp. 3-28. Springer,

[5]. Cysneiros Luiz Marcio, Vera Werneck, Juliana Amaral and Eric Yu. 2005. "Agent/goal Orientation versus Object Orientation for Requirements Engineering: A Practical Evaluation Using an Exemplar." Proc. of VIII Workshop in Requirements Engineering.

[6]. Luck Michael. May 2006. "Agent-based Computing" GEOconnexion International Magazine.

[7]. Uhrmacher Adelinde, Tyschler P, and Tyschler D. 1997. "Concepts of object-and agent-oriented simulation." Transactions of the Society for Computer Simulation 14.2: 59-67.

[8]. Praehofer H., Auernig F., Reisinger G., 1994. "STIMS - Modeling and Simulation Environment". Technical Report, STIMS-94-2, Institute of Systems Science, Johannes Kepler University Linz, Austria.

[9]. Tooli Amin Farahbakhsh, Asadi Javad. 2011. " Evaluating GAIAMethodology in Agent-Oriented Software Engineering". 5thSASTech 2011, Khavaran Higher-education Institute, Mashhad, Iran. May 12-14.

[10]. Odell James, 2002: "Journal of Object Technology", Published by ETH Zurich, Chair of Software Engineering ©JOT, vol. 1, no. 1, May-June 2002. Online at http://www.jot.fm

[11]. Silva, Viviane, 2003. "Taming agents and objects in software engineering." Software engineering for large-scale multi-agent systems : 103-136.

[12]. Russell Stuart and Norvig Peter, 1995. "Artificial Intelligence: A Modern Approach by, Prentice-Hall", Inc. 1995

[13]. Bratman 1987. "ME Intentions, Plans, and Practical Reason". Harvard University Press, Cambridge, MA

[14]. Rao AS and Georgeff MP. 1992. "An abstract architecture for rational agents". In Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge.

[15]. Abeck Sebastian, Koppel A, Seitz J, 1998. "A management architecture for multi-agent systems," Systems Management, 1998. Proceedings of the IEEE Third International Workshop on , vol., no., pp.133-138, 22-24 Apr 1998

[16]. Dong Hong, Chun Yi. 2000. Research on Mobile Agent Technology. Computer Science, 2000, 27(4):35-38.

[17]. Li Suhong, Chen Liwen, Li Guihong, 2009 "Overview of Application Research on Multi-Agent," Management and Service Science, 2009. MASS '09. International Conference on , vol., no., pp.1-4, 20-22 Sept.

[18]. Papazoglou M, Laufman P, Sellis K. 1992. "An organizational framework for cooperating intelligent information systems". Journal of Intelligent and Cooperative Information Systems, 1(1): 169-202.

[19]. Deloach Scott, Wood Mark, 2001 “Developing Multi-agent Systems with agent Tool”, in Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International Workshop, Boston, USA, July2000, published in LNCS, Vol. 1986, Springer Verlag, Berlin.

[20]. Bernon C., Camps V., Gleizes M. P. and Pi scard G. 2003. ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering; In: Lecture Notes in Computer Science Volume 2577, Springer Berlin He idelberg, ISSN: 0302-9743, pp. 156-169.

[21]. Henderson-Sellers, Brian & Giorgini, Paolo (ed). 2005. Agent-oriented Methodologies 1ed: Idea Group Inc, London, UK, ISBN 1-59140-581-5, p412.

[22]. Vera Maria B. Werneck, Rosa Maria E. Moreira Costa and Luiz Marcio Cysneiros, 2011. Modelling Multi-Agent System using Different Methodologies Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-176-3, InTech, DOI: 10.5772/14792. Link.

[23]. Danilo Rosa dos Santos, Marcelo Blois Ribeiro, Ricardo Melo Bastos. 2012 "A Comparative Study of Multi-Agent Systems Development Methodologies". Caixa Postal 90.619-900 – Porto Alegre – RS – Brazil.

[24]. Yu, E. (1995). “Modelling Strategic Relationships for Business Process Reengineering”. Ph.D. thesis. Dept. of Computer Science, University of Toronto.

[25]. Bastos, R. M., Ribeiro, M. B. (2004). “Modeling Agent-Oriented Information Systems for Business Processes”. In:

Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, Edinburgh, Scotland. 26th International Conference on Software Engineering – Workshop. United Kingdom: The IEE. p. 90-97.