Evaluation of Agent Oriented Software Engineering (AOSE) Methodologies-A review

1Sukhvir Singh, 2Prachi, 3Richa Setiya

123Department of CSE, NCCE, Israna, Panipat, India
2prachi_madaan@yahoo.co.in
3richasetiya@rediffmail.com

Abstract - In recent years, the agent technology has evolved rapidly along with a growing number of agent architectures, theories and languages. Computer science and engineering communities considered agent technology as one of the most important and active area of research and development. Agent oriented software engineering has numerous applications in different areas such as information management, space exploration, air traffic control, electronic commerce, business process management, defense simulation etc. A growing number of agents adopting software engineering methodologies have been proposed in recent years. The purpose of these methodologies is to provide models, methods, tools and techniques so that the development of software’s can be achieved in a systematic way. Even a large number of methodologies for agent-oriented software engineering are developed; a complete agent-oriented methodology for developing agent systems is still absent. In this paper, we explore the various applications of Agent-based systems categorized into different application domains.

Keywords - Multi agent system (MAS), Multiagent Systems Engineering (MaSE), Unified Modelling Language (UML), Agent Oriented Programming (AOP), Rational Unified Process (RUP)

I. INTRODUCTION

Agents are powerful technology with many important applications. Industrial software’s are complex in nature. They are typically characterized by a large number of components that have many interactions. Moreover, this complexity is not accidental: it is an innate property of the types of tasks for which software is used. The role of software engineering is therefore to provide structures and techniques that make it easier to handle this complexity. The advent of multi-agent systems has brought together many disciplines in an effort to build distributed applications, intelligent, and robust. They have given us a new way of looking at distributed systems and open a path to more robust intelligent applications. However, many of our traditional ways of thinking and designing software do not fit the multi agent paradigm. In recent years, there have been several attempts of creating tools and methodologies for building such systems. Unfortunately, many of the methods focused on single-agent architecture or have not gone to the level of detail necessary to adequately support the development of complex systems [1].

Advances in software engineering in the past two decades have been done through the development of increasingly powerful and natural high-level abstractions with which to model and develop complex systems. Procedural abstraction, abstract data types, and more recently, objects and components are all examples of such abstractions. It is our belief that agents represent a similar advance in abstraction that can be used by software developers to understand, model, and develop an important class of complex distributed systems, more naturally.

Much of current research related to intelligent agents has focused on the capabilities and structure of individual agents. However, to solve complex problems, these agents must work cooperatively with other agents in a heterogeneous environment. This is the area of multi agent systems. In multi agent systems, we are interested in the behavior of a coordinated system of individual agents to provide a system-level behavior.

The five different key agent-oriented methodologies: GAIA, MaSE, MESSAGE, Prometheus and Tropos have been compared by performing a feature analysis. This is carried out by studying the strengths and weaknesses of each methodology and combining their strengths for the development of new extensions. The work has been accomplished by an attribute-based evaluation framework. The four major areas of an agent-oriented methodology: modeling language, concepts, process and pragmatics have been addressed.

II. LITERATURE REVIEW

Literature review describes the methods and existing research approaches that are used systematically in the software engineering for requirement analysis and design. The methods described are widely analysed, establishing the scope and depth of work done to date, highlighting the current position of the achievements. Each of the individual methods is highly critical. The analysis also exposes the
tools and the present techniques within the stages of the methods. These tools and techniques are evaluated in order to reveal common situations and inconsistencies between the methods, tools and techniques.

To qualify as an agent, a software or hardware is often required to have properties such as autonomy, social ability, reactivity and proactivity. Other attributes that are sometimes required are mobility, veracity and rationality. A software system having all these properties is designed and programmed at a level of knowledge. Therefore, in the Agent Oriented Programming (AOP), we talk about mental states and beliefs instead of machine states, and also about plans and actions instead of the procedures and methods of communication, and also about negotiation and social ability rather than interaction and I/O functionality of goals, desires etc.

In particular, there are five major techniques were identified in relation to the development of agent oriented software GAIA, Multiagent Systems Engineering (MaSE), MESSAGE, Prometheus and TROPOS. The GAIA, MaSE and TROPOS are the only methodologies which allow us to exploit all the flexibility provided by Agent Oriented Programming and considered as a useful approach to follow in software development.

A. GAIA

The Gaia methodology is both general, which is applicable to a wide range of multi-agent systems (MAS), and comprehensive, which covers both the macro (social) and micro (agent) aspects of systems. Gaia is based on the idea of a multi-agent system as a computational organization consisting of various interacting roles as in [2]. GAIA has some limitations:

- Being an actor, GAIA does not attempt to deal with systems, explicitly, where agents cannot share common goals.
- Within GAIA, the representation of the cooperation protocols between the agents is currently poor to some extent.
- Open or Dynamic systems in which system components may enter and leave at run time and can be composed of entities that a designer would not be aware of at design time have been reported as a difficult class of system.
- GAIA was not designed with any particular standard for communication between the agents in mind. However, in the case of large-scale industrial collection of such standards, GAIA may be useful to adapt our methodology to be compatible with such standards.

B. Multiagent Systems Engineering (MaSE)

MaSE is an agent-oriented software engineering methodology that is an extension of Object-Oriented approach. MaSE treats agents as a simple software processes that interact with each other to meet the overall goals of the system. MaSE allows all system components equally, regardless, they have the intelligence or not. MaSE is built according to the application of existing techniques for the Object-Oriented analysis and design. As a software engineering methodology, the main objective of MaSE is to provide a complete life cycle methodology to help system developers to design and develop a multi-agent system as in [3]. It has certain limitations:

- Some of the software applications are closed and that all the external interfaces are encapsulated by an agent who participates in the system communication protocols.
- The methodology does not consider dynamic systems in which agents can be created, destroyed or moved during execution.
- Conversations between the agents are believed to be one-to-one, opposed to multicast. However, a number of spare point-to-point messages are used to fulfill the request for multicast.
- The systems with MaSE would not be considered very large.

C. MESSAGE

The methodology covers MAS (Multi Agent System) analysis and design and is intended for use in mainstream software engineering departments. The MESSAGE notation extends the UML with agent knowledge-level concepts and diagrams with notations for viewing them. The MESSAGE analysis and design process is based on the Rational Unified Process-based (RUP). Experimenting with the methodology shows that the analysis and design of multi-agent systems that is flexible and is supported on certain types of changes in a heterogeneous and dynamic environment.

In particular, existing approach does not adequately capture an agent flexible, autonomous behavior problem solving, the richness of the interactions of an agent, and the complexity of organizational.

D. PROMETHEUS

Prometheus is a methodology for developing intelligent agent systems. It has been developed in collaboration with the agent-oriented software and a modified version of Prometheus modeling language has been partially implemented in JACK. Intelligent JACK as a visual modeling tool design and architectural plans.

E. TROPOS

Tropos is based on two key ideas. First, the notion of agent and related mentalistic notions all (eg, goals and plans) are used in all phases of software development, from early analysis to actual implementation. Second, Tropos covers also the early stages of requirements analysis, allowing a deeper understanding of the environment in which the software must operate, and the type of interactions that must occur between software and human agents as in [4]. It has some limitations:

- This methodology is not intended to any type of software. For system software (eg compilers) or embedded software, the operating environment of the system-to-be is an engineering artifact, without identifiable actors.
- The Tropos methodology in its current form is not suitable for sophisticated software agents for an extended reasoning mechanisms for plans, objectives and bargaining.
III. COMPARISON OF DIFFERENT METHODOLOGIES

There have been several types of comparisons made previously by many researchers and software engineers; these comparisons are based on certain different criteria like criteria related to the process, the steps and techniques related criteria, measures and criteria for usability, model related and “concepts” related criteria, the comparison with respect to the model related criteria and comparison with respect to the support related criteria. All these different comparisons cover almost all the features of these methodologies such as application development life cycle support, the coverage of life cycle, development approach, the type of application domain, the nature of the agent, ease of understanding of the stages of development, etc. Ironically, the “best” method cannot be judged because all these methodologies are application oriented and none of them can be considered as a perfect model or framework for all types of agent-based applications. The analysis framework is based on four criteria:

A. Model Related Criteria

The criteria related to the model examine the capabilities and characteristics of the methodology models and components of notation. The ownership concept is divided into three further sub-sections: Internal, the social and technical properties. It constitutes

i) AUTONOMY, which states that agents can execute, operate and can be self-determining their own, without any direct human intervention. Agents must have an inherent check on their internal state that is dynamic in nature and may be modified by taking inputs from other agents in the environment.

ii) REACTIVITY; which states that agents must respond consistently with respect to the changes occurring in the environment. The changes are caused by other agents in the environment.

iii) CONCURRENCY states that agents must interact with other agents simultaneously for more than one goal.

iv) PRO-ACTIVENESS states that agents must keep track of their goals change over time. The objectives may evolve due to changes in the environment.

v) ENVIRONMENT BELIEF, which states that agents should receive input from the environment, act accordingly and then can provide an output for the environment, which can be used by others working in the environment.

vi) COOPERATIVE BEHAVIOR, which states that agents can ask, answer, deny and even negotiate with others to meet their individual goals and objectives of the system.

vii) COMMUNICATION ABILITY, which states that agents can communicate directly, transitively, one direction (one to one) or as a multi-directional transmission system like broadcast systems.

viii) ACP (Agent Communication Protocol) states that different agents communicate with each other by using message passing as a mean of communication. These messages can be two fold as well. A valid message sequence required to achieve the goal.

ix) ACL (Agent Communication Language) provides a mean to agent for exchanging knowledge and information between them. Using ACL, agents can pass messages in the network using low level and high level protocols.

x) COMPLETENESS & EXPRESSIVENESS, to model the system from the architectural point of view and from the unit point of view.

xi) CONSISTENCY is the property requires no contradiction between different models.

xii) MODEL REUSABILITY is the property of any component to be reused by other system with less or even no modification.

xiii) ABSTRACTION & MODULARITY, abstraction refers to the ability of Agent Oriented Methodologies to produce models at various levels of detail. Modularity is the property of dividing the system into small manageable chunks.

B. Technique Related Criteria

This approach deal with the assessment of the techniques of the methodology’s for carrying out development measures and / or to produce models and components notation.

a. AVAILABILITY OF TECHNIQUES & HEURISTICS, This is the property of AOM to provide techniques for carrying out each step of the process. Techniques for producing each model and the components of the notation.

b. TECHNIQUE USABILITY, AOM should provide a systematic structure to follow to develop a system model.

c. EASE OF UNDERSTANDING, the entries provided by the AOM should be easy to learn and remember different types of users. This requires the inclusion of symbols and notations that are familiar to users.

C. Process Related Criteria

This criterion examines the applicability of the AOM, the measures provided for the development process and the development approach followed by the AOM.

- DEVELOPMENT LIFE CYCLE, This deals with the development context (model, prototype model, iterative enhancement model etc.) supported by the AOM.
DEVELOPMENT PROCESS STEPS. This criterion assesses the tasks and activities specified by the AOM to the development process.

VERIFICATION & VALIDATION SUPPORT. Are we building the right system? Are we building it right? Both questions must be answered in order to have a clear idea about the accuracy of the models developed and specified requirements.

REFINABILITY, a simplified sequence of steps must be provided by the methodology for adding new details to the existing model. Refinement allows developers to make the necessary changes in the gradual stages of design development in an easy and simplified manner.

D. Support Features Related Criteria

These are “add on” features provided by any of these methodologies. This criterion assesses several additional features provided by any AOM. It includes CASE tools to support dynamic and open systems that you add and remove agents. Support for mobile agents and combination of conventional objects in MAS also included in the pattern of support.

SOFTWARE & METHODOLOGICAL SUPPORT. This criterion assesses the availability of various support tools such as CASE tools and development libraries for the development of MAS.

OPEN SYSTEM DEVELOPMENT SUPPORT.

Multi-agent systems are dynamic in nature. Several agents interact to achieve its objective. In a dynamic open system, agents can be added or removed to and from the system at any point in time. This criterion assesses the support provided by an AOM to develop the open agent based system.

In the next paper, we will compare the characteristics of five main Agent Oriented Methodologies (AOM) Gaia, MaSE, Prometheus, and Tropos & MESSAGE to the previous framework to analyze the mentioned features.

IV. CONCLUSION

As agent orientation approaches represent an emerging paradigm in software engineering, Agent paradigms are strongly demanded by the organization where software applications are large, complex and in different domains. Thus, the availability of agent methodologies that support software engineers in the development of agent-based systems is very important. In recent years, there have been a growing number of methodologies developed for agent-oriented software engineering. However, none of them is mature and complete enough to fully support the needs of industry for the development of agent-based systems.

REFERENCES
