An Agent Based Software Approach towards Building Complex Systems

Latika Kharb [1], Enes Sukic [2]

[1] *Jagan Institute of Management Studies, Rohini, New Delhi, India*
[2] *Faculty of Electronic Engineering, University of Nis, Nis 18000, Serbia*

***Abstract* – Agent-oriented techniques represent an exciting new means of analyzing, designing and building complex software systems. They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled. Yet, to date, there have been few serious attempts to cast agent systems as a software engineering paradigm. This paper seeks to rectify this omission. Specifically, points to be argued include:firstly, the conceptual apparatus of agent-oriented systems is well-suited to building software solutions for complex systems and secondly, agent-oriented approaches represent a genuine advance over the current state of the art for engineering complex systems. Following on from this view, the major issues raised by adopting an agent-oriented approach to software engineering are highlighted and discussed in this paper.**

***Keywords* –** Agent systems, software engineering, complex systems, software solutions.

1. Introduction

Over time, due to increased product functionalities, softwareprojects have become more and more complex and along withincreasing work completion pressures, the software projects arerequired to be accomplished in lesser amount of time but withfewer people[1].Designing and building high quality industry based software is a difficult task. Indeed, it has been verified that the industrial development projects are amongst the most complex construction tasks undertaken by humans. To handle this level of criticality, a wide range of software engineering paradigms have been devised (e.g., procedural programming, structured programming, declarative programming, object-oriented programming, design patterns, application frameworks and component-ware).Each successive development either claims to make the software development process easier or to extend the complexity of applications that can feasibly be built.But recently, with the high rate of increase in complexity of projects associated with software engineering, agent concepts have been considered as a new paradigm for handling complex systems.Agile methods are built on the assumption that the world isunpredictable and therefore aims at being adaptive, flexible and responsive, while traditionalmethods aims at optimizing the development through a well-planned and formalized process[2].By adopting an agent-oriented approach to software engineering means decomposing the problem into multiple, interacting, autonomous components (agents) that have particular objectives to achieve.

Yet despite this, a number of fundamental questions aboutthe nature and the use of the agent-oriented approach remain unanswered [3].Using agents as the basic building blocks to construct complicated software systems was first suggested by Shoham in 1993 [4].However, Agent Technology originated from artificial intelligence research and can be traced back to the actor model by Hewitt [5]of 1970.

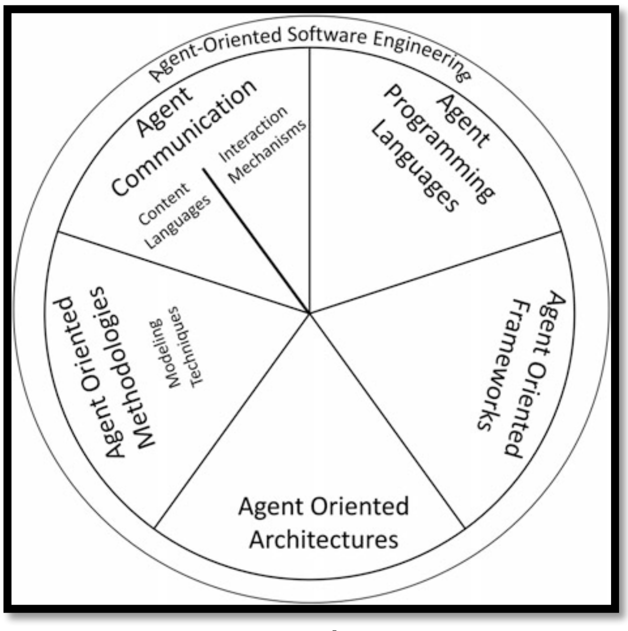
Agent-Oriented Software Engineering
Agent Communication
Content Languages
Interaction Mechanisms
Agent Programming Languages
Agent Oriented Frameworks
Agent Oriented Architectures
Agent Oriented Methodologies
Modeling Techniques

Fig 1: AOSE thematic map

Broadly speaking, an agent can be defined as:

- *Agent is a business process of developing software, equipped with distinct concepts and modeling tools, in which the key abstraction used in its concepts is that of an agent.*
- *An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.*

***User Satisfaction = Compliant Product + Good Quality + Delivery within Schedule*[6]**

There are a number of points about this definition that require further explanation. Agents are:
(i) clearly identifiable problem solving entities having well-defined boundaries and interfaces;
(ii) designed to fulfil a specific role—they have particular objectives to achieve;
(iii) autonomous—they have control both over their internal state and over their own behaviour;
(iv) capable of exhibiting flexible problem solving behaviour—they need to be reactive/ proactive.

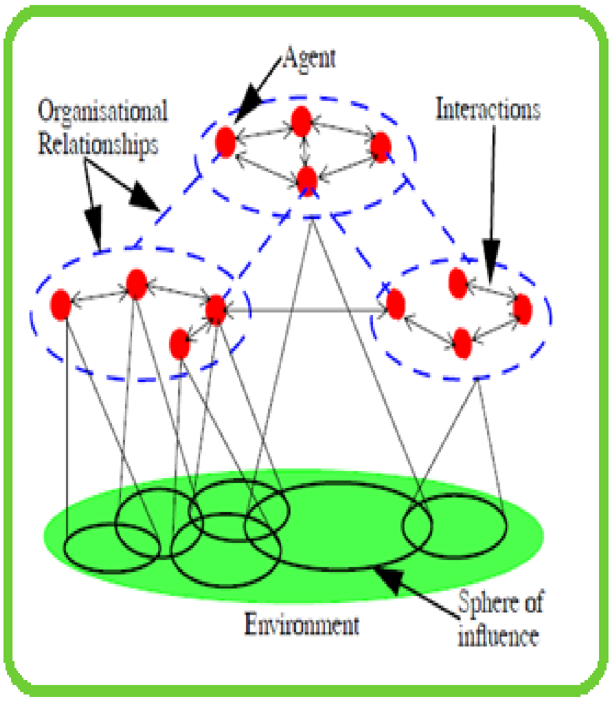

Fig 2:Canonical View of Agent Based Systems

When adopting an agent-oriented view of the world, it soon becomes apparent that a single agentis insufficient. Most problems require or involve multiple agents: to represent the decentralizednature of the problem:having multiple perspectives. Moreover, the agents will need to interact with one another, either to achieve their individual objectives or else to manage the dependencies that ensue from being situated in a common environment.

2. The Agent-Oriented Software Terminology

The agent based systems represent a promising approach for engineering complex systems. In particular, we explore agent based systems in three broad terms :

- *Specification*
- *Implementation*
- *Verification*

- **Specification**

In this, we consider all the problems ofspecifying an agent system, their requirements for a framework development etc.

- **Implementation**

The next issue we consider is:once specification is done, we are now ready to implement a system i.e. moving from specification to a computational system.

- **Verification**

Once a concrete system is developed, we need to check that if the system is correct with respect to our original specification. This process is known as verification.

3. Properties of Agent Based Systems

By an *agent-based system*, we mean a system that has the following properties:

- **Autonomy**: agents work in autonomy i.e. they encapsulate some state and make decisions about what to do based on this state, without the direct intervention of humans or others;
- **Reactivity**: agents are *situated* in an environment and are able to *perceive* this environment to respond in a timely fashion to the changes that occur in it;
- **Pro-activeness**: agents do not simply act in response to their environment, they exhibit goal-directed behavior;
- **Social ability**: agents interact with other agents via some kind of *agent-communication language*in order to achieve their goals.

Consider an example to understand the properties of agent based systems in a better way.An agent based automatic pilot aircraft system is designed to fulfill:

- **Proactiveness:**It's the property to plan how to safely land aircraft at an airport.
- **Reactiveness:**It's the property to foresee unseen circumstances and make system ready to react to them.
- **Social Ability:**It's the property that depictscooperation between air craft controllers and auto pilot system.
- **Autonomy:** All above mentioned properties whencompleted successfully; lead to property of autonomy.

4. Characteristics of complex systems

When we have to build complex systems, following characteristics of complex systems need to be enumerated [7]:

• Complexity is composed of inter-related subsystems, which arehierarchical innature. Moreover, these relationships are not static and they often vary over time.

• The choice of which components in the system are primitive is relatively arbitrary and is defined by the engineer's aims and objectives.

• Complex systems will evolve from simple systems more rapidly if there are stable intermediate forms, than if there are not.

• It is possible to distinguish between the interactions among sub-systems and the interactions within sub-systems. Moreover, although many of these interactions can be predicted at design time, some cannot.

5. Tackling complexity for Building Complex Systems

The most compelling argument that can be made for adopting an agent-oriented approach to software development is to have a set of quantitative data that showed the superiority of the agent-based approach over a range of other techniques. However such data does not exist. Hence arguments must be qualitative in nature.Booch [8] identifies three tools for tackling complexity in software:

• **Decomposition:** The most basic technique for tackling large problems is to divide them into smaller, more manageable parts which can be solved easily in isolation and it helps to tackle complexity because it limits the designer's scopebecause only a small portion of the problem needs to be solved at a time.

• Abstraction: The process of defining a simplified model of the system that emphasises some of the details or properties, while suppressing others. This technique is very effective as it limits the designer's attention, so that it can be focused on the salient aspects of the problem instead of diverting towards less relevant details.

• **Organisation:** The ability to specify organisational relationships can help designers to tackle complexity in two ways. Firstly, the individual components of a sub-system can be treated as a single unit by the parent system. Secondly, a number of components may work together to provide a particular functionality.

6. Strengths of Agent-oriented Methodologies

Agent-oriented methodologies strengths could be considered in two different ways:

- **Inclusion of other paradigms capabilities as well as presentation of more abilities:**

AOSE paradigm includes all the capabilities of other existing paradigms and also possesses more capabilitieslike – more features, more goal- oriented, good service performers and many more.

- **Suitability with new software development requirements:**

As mentioned before, due to the complexity of software development process, wide range of software engineering paradigms has been devised. But recently, with the high rate of increase in complexity of projects in software engineering, agent basedconceptshave made advancements

7. Weakness of Agent-oriented Methodologies

Agent-oriented methodologies weaknesses could be considered in three different ways:

- **Lack of Agent-oriented programming languages:**
  Although due to the complexity of software development process, agent based concepts, have achieved advancements.However, industry is reluctant to adopt a new paradigm as it seems impossible to implement these ideas in a currently acceptable, commercially viable programming language.

- **Lack of knowledge about existence & advantages of agent-orientation:**
  The benefits of agent technology must be declared by introducing the cases where agent based paradigm succeeds and other existing paradigms fail.

- **High cost of agent based system acquisition:**
  Being a new paradigm, it's acquisition by software development organizations requires a high cost for training the development team.

8. Comparisons& Mapping amongst Agent, Object and Component Based Systems

In this section, we have highlighted the comparison study of Object and Component based approaches with Agent-Based approach with respect to some characteristic features.

Table 1: Agents versus Objects and Components

Point of Differences	**Object-based Approaches**	**Component-based Approaches**
Passive in nature	Yes	No
Do not encapsulate behavior activation	Yes	No
Require adequate set of concepts/ mechanisms for modeling of system	Yes	No
Provide minimal support for structuring collectives	Yes	No

Table 2: Comparing Object & Agent-based Approaches

Point of Differences	**Object-based Approaches**	**Agent-based Approaches**
Passive in nature.	Yes	No
Do not encapsulate behavior activation	Yes	No
Fails to provide an adequate set of concepts / mechanisms for modeling complex systems	Yes	No
Minimal support for specifying/ managing organizational relationships	Yes	No

In the last few years, software development research has focused on methods and approaches that work towards developing soft-ware systems by integrating already developed components [9]. Often, software engineers use object-oriented programming to implement agent systems. There exist similarities between object oriented and agent oriented paradigms. A technology that is closely related to that of object-oriented systems is component-based software [10]. There are a number of motivations for component-based software, but arguably the single most important feature of component systems is software reuse.Everyday software development projects develop all software components from scratch. Researchers have now developed methods that permit building software from pre-built components.
Like components, agents are typically self-contained computational entities, that do not need to be deployed along with other components in order to realise the services they provide. Also, agents are often equipped with "metalevel reasoning" ability i.e. they are able to respond to requests for information about the services they provide. So, with respect to agent-oriented approach; it is possible for proponents of object-oriented systems/component oriented systems, or any other programming paradigm to claim that such mechanisms can be implemented using agent based technique.

9. Conclusion

In this paper, we've described the basic terminologyof agent based systems, their properties, as well as their advantages and weaknesses. A comparative analysis of existing paradigms (object based and component based) with newer paradigm (agent based) is also presented in this paper. As we've discussed that a very little awareness is present with respect to this paradigm; so our concern to write this article is to generate a technical awareness amongst researchers and industry people about its existence. To briefly conclude here, we can write that today agent based software approach for building complex systems is at an initial stage. A lot of work has to be done to make this engineering paradigm a popular one, and make the industry and researchers realize it's worth and start development of complex systems with agent based approach.The paper concludes by setting out some issues and open problems for future research through comparative study.

References

[1] Dr. Latika Kharb, Proposing a Comprehensive Software Metrics for Process Efficiency, International Journal of Scientific & Engineering Research, Volume 5, Issue 9, September-2014.

[2] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Information and Software Technology, vol. 50, pp. 833-859, 2008.

[3] Nicholas R. Jennings, On agent-based software engineering, Artificial Intelligence (Elsevier Science), pp 277–296, 2000.

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

[5] Hewitt, C. 1990, Toward open Information Systems Semantics, Proceedings of 10th International Workshop on Distributed Artificial Intelligence, Technical Report, ACT-AI-355-90, MCC, Austin, Texas.

[6] Latika Kharb et al, Reliable Software Development with Proposed Quality Oriented Software Testing Metrics, Int. J. Comp. Tech. Appl., Vol 2(4), JULY-AUGUST 2011

[7] Simon, 1996]: H. A. Simon (1996) “The sciences of the artificial” MIT Press.

[8] Booch [1994: G. Booch (1994) “Object-oriented analysis and design with applications” Addison Wesley

[9] Latika Kharb et al, Complexity Metrics for Component-Oriented Software Systems, ACM SIGSOFT Software Engineering Notes, Volume 33 Number 2, March 2008.

[10] C. A. Szyperski (1997) “Component software: beyond object-oriented programming” Addison Wesley.