COMPARING AGENT –ORIENTED PROGRAMMING VERSUS OBJECT-ORIENTED PROGRAMMING

Rula K.Al-Azawi

Gulf College affiliated with Staffordshire University-UK
Muscat/Oman
E-mail: rulaalazzawi@yahoo.com

Aladdin Ayesh

DMU University,UK
E-mail: aayesh@dmu.ac.uk

Abstract

The main goal of this paper is to overview the rapidly evolving area of agent oriented programming by comparing it with object oriented programming. As a first step, we will explain the meaning of agent and MAS. Next we present a typology of agents and the demonstrators of the various agent types in the typology. We have provided an overview on general issues such as the description of object oriented programming in general, and the agent oriented programming in details. Furthermore, it also provides an overview on the comparison between agent and objects and the agent oriented programming with object oriented programming.

Keywords - Agent –oriented programming, object –oriented programming ,multi-agent system.

1 INTRODUCTION

AGENT oriented programming is one of the most important areas of research and development to have emerged in information technology in recent years, which can be viewed as a specialization of object-oriented programming.

The state of an agent consists of four components; firstly beliefs that agent have information about their environment with may be incomplete or in correct; secondly goals, that agent will try to achieve; thirdly actions that agent perform and the effects of those actions; finally ongoing interaction that how agent interact with each other's and their environment over time. For this reason the state of an agent is called its mental state. The mental state of agents is described formally in an extension of standard epistemic logics: besides temporal the knowledge and belief operators [4].

This paper discusses both the similarities and differences between objects and agents and lets you decide which viewpoint you want to choose. Whichever viewpoint you choose, we hope you will find that the agent-based way of thinking brings a useful and important perspective for system development.

The objects could be used to support the agent-based approach, just like any modular languages. We provide another way of thinking about systems and their implementation. Agents, then, are an evolution rather than a revolution [1].

This paper will organized as follows, section 2 outlines the main characteristics of agent and Multi-Agent System (MAS); section 3 presents an overview of Agent Oriented Programming(AOP); section 4 presents an overview of Object Oriented Programming(OOP); section 5 presents in details the comparison between AOP various OOP; finally conclusion are presented

2 AGENTS AND MULTI-AGENT SYSTEM

This section outlined the main characteristics of agents and MAS and different way to definite agents followed by describe of MAS.

The basic concepts are presented from a very general point-of-view. The main aspect of this introduction is its use of a very general notation scheme for the description of agents in order to be independent of any particular agent school. [10]

2.1 What is an agent?

According to Jennings [12], an agent is an object capable of displaying the following:

Autonomy: The ability to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

- (Structural) Reactivity: The ability to perceive the environment1, and respond regularly to changes that occur in it.
- Social Ability: The ability to interact with other agents (and possibly humans) via some kind of agent-communication language.
- Pro-Activity: Goal The ability to exhibit goal-directed behavior by taking the initiative instead of just acting in response [7].

From the viewpoint of AI researchers an agent is a computer system that having the attributes mentioned above, is either conceptualized or implemented using concepts that are more usually applied to humans, such as knowledge, belief, intention, and obligation (and sometimes emotion) [5]. Some other attributes of agency are:

- Mobility: The ability to move around an electronic network.
- Veracity: The assumption of not communicating false information knowingly.
- Benevolence: The assumption of not having conflicting goals.
- Rationality: The assumption of acting with a view to achieve its goals, instead of preventing them.

2.2 Typology of Agents

In This section we attempts to place existing agents into different agent classes. A typology refers to the study of types of entities. There are several dimensions to classify existing software agents.

- Autonomy refers to the principle that agents can operate on their own without the need for human guidance, even though this would sometimes be invaluable. Hence agents have individual internal states and goals, and they act in such a manner as to meet its goals on behalf of its user.
- Cooperation with other agents is paramount: it is the reason for having multiple agents in the first place in contrast to having just one. In order to cooperate, agents need to possess a social ability
- Learn, for agent systems to be truly smart, they would have to learn as they react and/or interact with their external environment [8].

We use these three minimal characteristics in Figure 1 to derive four types of agents to include in the typology: collaborative agents, collaborative learning agents, interface agents and truly smart agents [7].

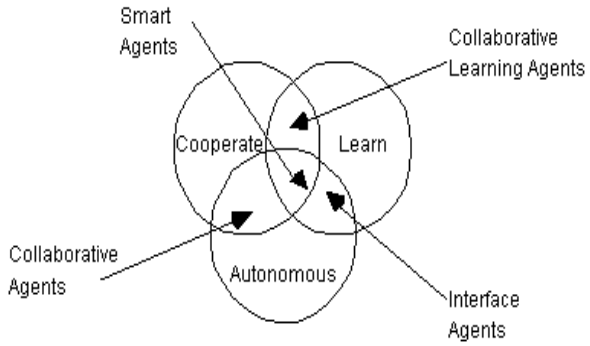

Figure1. A Part View of an Agent Typology

In principle, by combining all the previous parameters, we could have summarizes of these types as shown in Figure2.

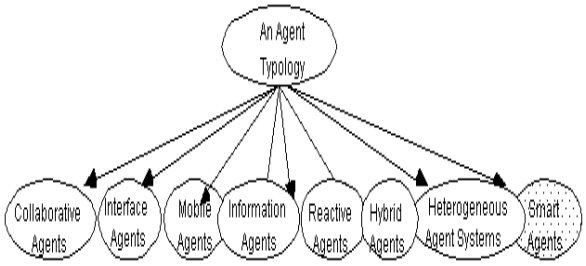

Figure 2. A classification of software agents

2.3 Multi-agent System (MAS)

MAS are the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors. Some domains require MAS. In particular, if there are different people or organizations with different (possibly conflicting) goals and proprietary information, then MAS is needed to handle their interactions.

MASs differ from single-agent systems in that several agents exist which model each other's goals and actions. In the fully general multi-agent scenario, there may be direct interaction among agents (communication). Although this interaction could be viewed as environmental stimuli, we present inter-agent communication as being separate from the environment.

From an individual agent's perspective, MAS differ from single-agent systems most significantly in that the environment's dynamics can be affected by other agents. In addition to the uncertainty that may be inherent in the domain, other agents intentionally affect the environment in unpredictable ways. Thus, all MAS can be viewed as having dynamic environments [13].

2.4 Agent Applications

Current and potential applications of agent technology include [14];

Area	Example
Industrial	Manufacturing, Process Control, Telecommunications, Air Traffic Control, and Transport System
Commercial	Information Management, E-Commerce, and Business Process Management
Entertainment	Games, Interactive Theatre, and Cinema
Medical	Patient Monitoring, Health Care

3 OVERVIEW OF THE AOP

AOP is a programming paradigm introduced by Yoav Shoham in [4].The objective of Agent Oriented (AO) Technology is to build systems applicable to real world that can observe and act on changes in the environment. Such systems must be able to behave rationally and autonomously in completion of their designated tasks. AO technology is an approach for building complex real time distributed applications. This technology is built on belief that a computer system must be designed to exhibit rational goal directed behavior similar to that of a human being.

In AOP objects known as agents interact to achieve individual goals. Agents can be autonomous entities, deciding their next step without the interference of a user, or they can be controllable, serving as mediatory between user and another agent. AOP programming is performed at abstract level. The Agent-Oriented Software Engineering (AOSE) is being described as a new paradigm for the research field of Software Engineering. But in order to become a new paradigm for the software industry, robust and easy-to-use methodologies and tools have to be developed [9].

Shoham [4] proposed AOP system has three components. First a logical system for defining the mental state of agents, second Interpreted program language for programming agents, and the third is An “agentification” process, for compiling agent programs into lower level executable systems [4].

4 OVERVIEW OF THE OOP

OOP focuses on data. How data is modeled and manipulated when it is used with objects is fundamental to any OOP. The first thing that we need to become familiar with is an object. An object is a bundle of software that contains variables and methods. The objects which are used in computer programs will often be used to simulate the objects in real world. An object will hold its state in more than one variable. A variable can be defined as information that has been identified with a name. The object uses its methods and behaviors. A method is an action that is carried out by the object. It may also be called a function.

The main concepts for object oriented programming are Classes, Objects, and Methods. There are some of important features that are applied in object oriented programming. These are Inheritance, Abstraction, Polymorphism, and Encapsulation. The first concept is called encapsulation which means protecting the variables within the center of an object by surrounding it with methods. The second concept is called abstraction which it can be defined as the process in which an application will ignore the characteristics of a sub-group and work on a general level when it is needed. The third concept is called inheritance which means the ability of a new class to be created, from an existing class by extending it. The last concept, Polymorphisms, is a generic term that means 'many shapes'. More precisely, it means the ability to request that the same operations be performed by a wide range of different types of things [11].

5 AOP VERSUS OOP

In this section we follow the idea of Shoham [5] by using table 1 to discuss those differences between AOP and OOP in details.

Table 1
OOP versus AOP

	OOP	AOP
Basic unit	object	agent
Parameters defining state of basic unit	unconstrained	beliefs, commitments, capabilities, choices, …
Process of computation	message passing and response methods	message passing and response methods
Types of message	unconstrained	inform, request, offer, promise, decline, …
Constraints on methods	none	honesty, consistency, …

5.1 Basic unit

The points listed below shows differentiators between agents and objects:

- Agents may communicate using an Agent Communication Language ACL or ICL, whereas objects communicate via fixed method interfaces.
- Agents have the quality of volition. That is, using AI techniques, intelligent agents are able to judge their results, and then modify their behavior (and thus their own internal structure) to improve their perceived fitness.
- Objects are abstractions of things like invoices. Agents are abstractions of intelligent beings -- they are essentially anthropomorphic. Note that this does not mean that agents are intelligent in the human sense, only that they are modeled after an anthropomorphic architecture, with beliefs, desires, etc.

5.2 Parameters defining state of basic unit

AOP agents contain beliefs, commitments, choices, and the like and communicate with each other via a constrained set of speech type acts such as inform, request, promise, decline the state of the agent is called its mental state.

5.3 Process of computation

An object's message may request only one operation, and that operation may only be requested via a message formatted in a very exacting way. The OO message broker has the job of matching each message to exactly one method invocation for exactly one object.

Agent-based communication can also use the method invocation of OO. However, the demands that many agent applications place on message content are richer than those commonly used by object technology. While agent communication languages (ACL) are formal and unambiguous, their format and content vary greatly. In short, an agent message could consist of a character string whose form can vary yet obeys a formal syntax, while the conventional OO method must contain parameters whose number and sequence are fixed..

5.4 Type of message

Agents are commonly regarded as autonomous entities, because they can watch out for their own set of internal responsibilities. Furthermore, agents are interactive entities that are capable of using rich forms of messages. These messages can support method invocation—as well as informing the agents of particular events, asking something of the agent, or receiving a response to an earlier query. Lastly, because agents are autonomous they can initiate interaction and respond to a message in any way they choose. In other words, agents can be thought of as objects that can say "No"—as well as "Go." Due to the interactive and autonomous nature of agents, little or no integration is required to physically launch an application.

Objects, on the other hand, are conventionally passive—with their methods being invoked under a caller's thread of control. The term autonomy barely applies to an entity whose invocation depends solely on other components in the system [6].

5.5 Constrain on method

Usually, object classes are designed to be predictable in order to facilitate buying and selling reusable components. Agents are commonly designed to determine their behavior based on individual goals and states, as well as the states of ongoing conversations with other agents. While OO implementations can be developed to include nondeterministic behavior, this is common in agent-based thinking.

Agent behaviour can also be unpredictable because of the following. First, an agent's knowledge can be represented in a manner that is not easily translated into a set of attributes. Even if an agent's state

were publicly available, it may be difficult to decipher or understand. Second, the requested behaviors that an agent performs may not even be known within an active system. This is a clear distinction from object systems, because current OO languages only let you ask an object what interfaces it supports. Since the programmer needs to have some idea what interface to ask for, this makes coding difficult. In OO, there is no provision in current languages for an object to "advertise" its interfaces. In contrast, an agent can employ other mechanisms, such as publish/subscribe, protocol registration, “yellow page” and “white page” directories. Another common mechanism provides specialized broker agents to which other agents can make themselves known for various purposes but are otherwise unlisted to the rest of the agent population. Lastly, the underlying agent communication model is usually asynchronous. This means that there is no predefined flow of control from one agent to another. An agent may autonomously initiate internal or external behavior at anytime, not just when it is sent a message [3]. There is no predefined flow of control from one agent to another. An agent may autonomously initiate internal or external behavior at anytime, not just when it is sent a message [3].

6 CONCLUSION

Agents employ some of the mechanisms and philosophies used by objects. In fact, many software developers strongly advocate composing agents from objects—building the infrastructure for agent-based systems on top of the kind of support systems used for OO software systems.

In MAS, an additional layer of software components may be naturally expressed as objects and collections of objects. This is the underlying infrastructure that embodies the support for agents composed of object parts.

The difference here does not mean bad or good—it is only different. In the end, you might conclude that agents are really just objects or that agent and objects are different but can peacefully coexist and even support one another in the same system. Either way, the agent-based way of thinking brings with it a useful and important perspective for system development [2].

References

[1] J.Odell , "Objects and Agents: How do they differ? ", DRAFT 2.2 2,Copyright James Odell ,1999.

[2] D.Levine, "Relationship between Agent and Object Technologies" in Odell, James, ed., *Agent Technology Green Paper*, OMG Agent Work Group, 1999.

[3] M.Wooldridge, N.R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design," *Autonomous Agents and Multi-Agent Systems,* forthcoming, 1999.

[4] Y.Shoham, "Agent-Oriented Programming"(Technical Report STAN-CS-90-1335). Stanford University: Computer Science Department, UK, 1990.

[5] Y.Shoham," Agent-oriented Programming". *Artificial Intelligence 60*, 60(1):51–92, 1993.

[6] H.V. Parunak, "'Go to the Ant': Engineering Principles from Natural Agent Systems," Annals of Operations Research, 75, 1997, pp. 69-101.

[7] H.S. Nwana ," Software Agents: An Overview" ,Intelligent Systems Research , Advanced Applications & Technology Dep. , Cambridge university U.K. Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996.

[8] M.Wooldridge & Jennings, N., "Intelligent Agents", Lecture Notes in Artificial Intelligence 890, Heidelberg: Springer Verlag, 1995.

[9] L.Foner, "What is an Agent, Anyway? A Sociological Case Study", Agents Memo 93-01, MIT Media Lab, Cambridge, MA, 1993.

[10] T.Wittig," An Architecture for Multi-Agent Systems", London: Ellis Horwood, 1992.

[11] P. Leahy, "Introduction to Object-Oriented Programming", http://java.about.com/od/objectorientedprogramming Object-Oriented_Programming.htm

[12] N.R.Jennings, "On Agent-Based Software Engineering. _Artificial Intelligence", vol. 117, pp.277-296, 2000

[13] P.Stone_ and M.Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective", In Autonomous Robotics volume 8, number 3. July, 2000.

[14] N.R. Jennings, K.P. Sycara, and M. Wooldridge. "A Roadmap of Agent Research and Development",Journal of Autonomous Agents and Multi-Agent Systems, 1(1):7–36, July 1998.