Different Views of Software Agent Paradigm

Anand Kumar Pandey
Assistant Professor, Deptt. Of MCA
ITM University, Gwalior - 474001
e-mail: anandpandey.ind@gmail.com

Rashmi Pandey
Assistant Professor, Deptt. Of MCA
ITM University, Gwalior - 474001
e-mail: reshu1807@gmail.com

Abstract: Agent modelling in software engineering is a relatively young area, and there are, as yet, no standard methodologies, development tools, or software architectures. Although our work is emphasized on to represent the agent specifications using different views, they are not oriented for software engineering in terms of providing a modeling notation that directly supports software development. There are, however, some software frameworks using different views of software agents, that are beginning to appear, and in this paper we describe one of them that use agent-oriented technclogy and which are interesting fields for the software development. Agent-oriented approaches can significantly enhance our ability to model, design and build complex (distributed) software systems.

Keywords: Multiagent System, Paradigm, Framework, Reactivity.

1. Introduction: Now a days the agent paradigm and its different views gaining popularity because it helps to system to bring efficiency, intelligence, accuracy and autonomy to software systems. The behavior and activity of an software agent (or the results of executing a method) may be different at different times (and may be deterministic or non-deterministic). Here we discuses the new approach of agent-oriented software engineering with their framework. Some applications that can be directly benefit from an agent-oriented design or framework are typically structured as multiagent systems, which are usually defined as a concurrent system based on the notion of autonomous, reactive, and internally-motivated agents in a decentralized environment [3]. Now a days, building a system using current technology is a complex and difficult task because the system decomposition forces the developer to deal with relatively low-level concepts.

2. Why are Software Agents important?: Now a days software agents are probably the fastest growing area of Information Technology (IT). A software agent is a computer program that is designed to assist a user both directly and indirectly. We define Software Agents as an autonomous entity driven by beliefs, goals, capabilities, plans and a number of agency properties, such as autonomy, adaptation, interaction, learning and mobility [2]. Software developers and system designers use high-level abstractions in building complex software for one reason - to manage complexity. An abstraction focuses on the important and essential properties of a problem and hides the incidental components of that problem. Agents provide a new way of managing complexity because they provide a new way of describing a complex system or process. Using agents, it is easy to define a system in terms of agent-mediated processes.

"An agent is an entity that:

- Acts on behalf of others in an autonomous fashion.
- Performs its actions in some level of proclivity and reactivity.
- It exhibits some levels of the other key attributes of learning, co-operation, and mobility."

Graesser classifies agents on the basis of the properties they possess (Franklin, S., and Graesser, A., 1996). The major properties are reactiveness, autonomy, goal-driven, temporal continuity, communicative, learning, mobility, flexibility, etc [1]. Every agent possesses first four properties. **Russell & Norvig** classify agents into four categories: Simple Reflex Agents, Reflex Agent with state, Goal-based Agents and Utility-based Agents. According to the **Brustoloni's** classification, agents have been categorized as Regulation Agent, Planning Agent and Adaptive Agent.

J. Alfredo Sánchez categorizes agents on the basis of their use in different domains, i.e. Programmer Agents, Network Agents and User Agents (J. Alfredo Sánchez, 1997) [3]. After study of various characteristics and classification of software agents we propose following classification:

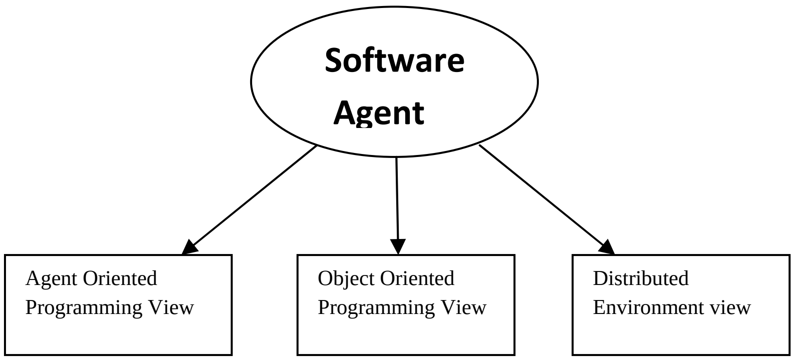
Software Agent
Agent Oriented Programming View
Object Oriented Programming View
Distributed Environment view

Figure 1 : Classification of Software Agent

3. **Agent Oriented Programming View:** Agent-oriented programming is a programming paradigm introduced by Yoav Shoham in 1990. Agent-oriented programming view is the next expected evolution of object oriented programming. We can say that it can be viewed as an specialization of object-oriented programming. Here, I am presenting my own viewpoint on agent-oriented software engineering by relating it to other programming paradigms. A widely accepted software engineering principle is that a system should be composed of a set of independent modules, where each module hides the internal details of its processing activities and modules communicate through well-defined interfaces. From the above description, we can see that a agent oriented model essentially represents a module or an object rather than an abstraction of a set of similar objects [4]. we defined an approach to extend this model to support class modeling. The idea of this extension is to generate a unique object identifier.

Agent-oriented techniques can be applied during both the phases, such as design phase, where agent-based techniques can be used to model the problem domain with its abstraction and the system design, and during the implementation phase, where agent-oriented development tools would be used. The use of the agent based approach in both phases is a natural fit, but not required. Not to use them together, however, would fail to take advantage of the natural mapping from one phase to the other and would be like implementing an object-oriented design using a procedural programming language.

3.1 Framework for Agent Oriented Approach: A framework for agent-oriented software (Figure 2) offers a set of services that can be requested either by users or by other application frameworks. A framework is an environment composed of a supervisor and one or several teams of software agents. The services provided by a framework are performed by different teams set up by the framework. These teams are composed of agents selected from a bank of software agents. This bank contains several agents having different functionalities which are specific to the application to be developed and to the characteristics of the information systems to be interconnected.

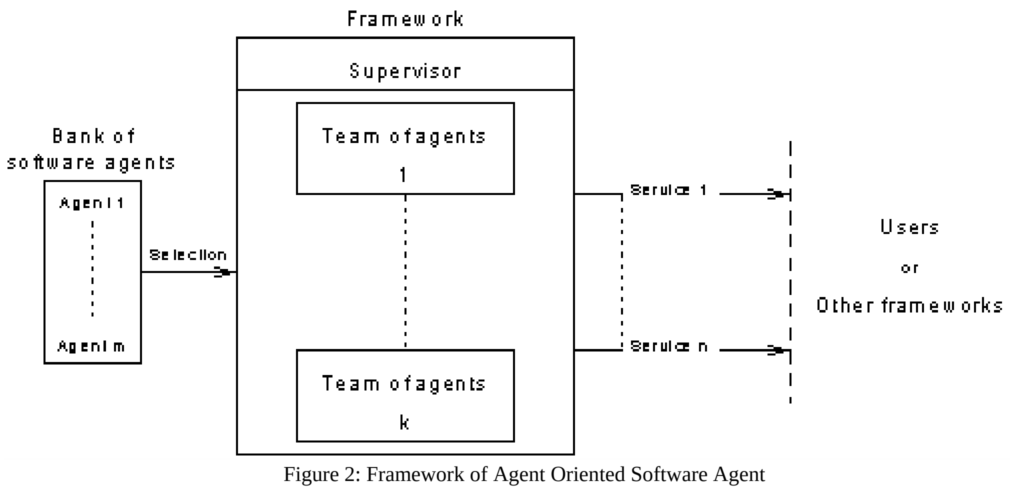

Figure 2: Framework of Agent Oriented Software Agent

"Teams of agents" are composed of several software agents and are structured differently according to their responsibilities in the framework. A team is properly classified and characterized by a team supervisor and a set of roles that agents must fulfill according to their capabilities. A "role" is identified by a list of parameters: role label, goals to achieve, and services offered by and required for the role. Considering these parameters, a framework selects appropriate agents from the bank of software agents. To fulfill their goals, agents need to cooperate and exchange information. Services provided by a framework satisfy specific users' needs such as browsing, query formulation, information retrieval, etc. When a service is invoked by a user, the framework's supervisor agent activates a scripting procedure, called a "realization scenario", which specifies the characteristics and the interaction protocols of the teams of agents that will perform the various operations required to carry out the service. Interactions can be defined between various kinds of components: frameworks, teams, and agents. According to that realization scenario, the framework supervisor creates teams that will play roles specified in the scenario. At their own level, team supervisor agents activate realization scenarios in order to coordinate the activities of their software agents.

4. **Object Oriented Programming View:** The object oriented paradigm is the framework in software engineering, influencing all effort in information science. It is one of the main objectives of the software engineering discipline. Object-oriented programming view of agents can be discussed through the studies of historical sources, standards and current research. Object models are different from other modeling techniques because they have merged the concept of variables and abstract data types into an abstract variable type: an object. Objects have identity, state, and behavior and object models are built out of systems of these objects. To make object modeling easier, there are concepts of type, inheritance, association, and possibly class [5]. We believe that object-oriented programming and agent technology can reduce not only the development time but also the complexity of implementing multi-agent systems.

Although we recognize that the object-oriented paradigm has some flaws related to design and implementation of multi-agent systems, we also believe that it is still the most practical programming language to implement the agent technology. This technology support the ability to build applications by selecting and assembling objects from libraries.

Components	**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.
No. of Methods	Many methods with variable parameters	One method with single parameters

Table 1: Comparison Table

The main goal of the OOP technology is to diminish the development time and reduce the complexity of implementing Software Agents. The design of this technology is inspired on the OOP Methodology, and it can introduce an easy implementation of all the models developed by the analysis and design phase.

5. **Distributed Environment View:** A Software Agent System (or Multi-Agent System (MAS)) is a computational environment (such as the Web or a grid computing environment overlay network) in which individual software agents interact with each other, sometime in a cooperative manner, sometimes in a competitive manner, and sometimes autonomously pursuing their individual goals. The distributed environment view is used to describing the environment that the system will exist in and need to interact with. It suggests representation of the environment model as distributed parts of data that the agents can perceive (read), effect, edit or consume (remove from environment).

Our main objective is to understand software agent approacg and topics in a distributed environment. Here we will produce a design using object-oriented patterns and notation for a distributed application and analyze its performance. Evaluate the design with respect to distributed environment issues. Propose a system test plan appropriate for a distributed application. Distributed environment view include web applications, other networks and wireless environments, as well as self-contained embedded system products involving multiple processors. In this paper, we will learn about software agent issues that exist because of the complexity of software running simultaneously and asynchronously on more than one processor. Topics: Specification, distributed design patterns, testing, distributed objects, security, time and global states, transaction coordination, data replication, and special context such as : web services, mobile computing, and multi-media.

6. Conclusion:

One of the most rapidly growing areas of interest for software engineering and distributed computing is that of distributed agent systems. Although there are several ways for implementations of agent systems available, formal frameworks for such systems are few. Formal methods in agent system specification and design can help to ensure robust and reliable products. In this paper, we introduced an agent-oriented model, which provides a foundation that is mature in terms of both existing theory and tool support. An example of an agent family in electronic commerce was used to illustrate the modeling approach. Models for a shopping agent, selling agent, buying agent, and retailer agent were presented, with emphasis on the characteristics of being autonomous, reactive and internally-motivated. Our agent-oriented models also provide a clean interface between agents and agents may communicate with each other by using decided way or tools.

Reference:

[1] A. Graesser and S. Franklin, “Is it an agent, or just a Program?” A Taxonomy for Autonomous Agents”. In Proceedings of the Third International Workshop On Agents Theories, Architecture, and Languages, Springer-Verlag, 1996.

[2] Garcia, A.; Lucena, C. J. An Aspect-Based Object-Oriented Model for Multi-Agent Systems. 2nd Advanced Separation of Concerns Workshop at ICSE'2001, May 2001.

[3] J. Alfredo Sánchez, 1997. “A Taxonomy of Agents” - Technical Report ICT-97-1 ICT Interactive and Cooperative Technologies Lab Universidad de las Américas-Puebla Department of Computer Systems Engineering A. P. 100 Cholula, Puebla 72820 México.

[4] Leonard Foner, "What's an Agent Anyway?", Agents Memo 93-01, Agents Group MIT Media Lab, 1993.

[5] W. Vongdoiwang, .D. N. Batanov. (2004). Similarities and Differences between Ontologies and Object Model. CCCT’05 proceeding 2004. Austin, Texas.